目錄

笨办法学 Linux 中文版

原书：Learn Linux The Hard Way (β version)

译者：飞龙

自豪地采用谷歌翻译

- 在线阅读
- PDF格式
- EPUB格式
- MOBI格式
- 代码仓库

赞助我

协议

练习 0：起步

> 原文：Exercise 0. The Setup
>
> 译者：飞龙
>
> 协议：CC BY-NC-SA 4.0
>
> 自豪地采用谷歌翻译

Windows，手动安装

非常长的指南

Windows，VirtualBox 虚拟机（`.ova`格式的预配置映像）

你需要什么

- VitualBox，虚拟机播放器。
- putty，终端模拟器。
- 预配置的 VirtualBox Debian 映像。

这样做

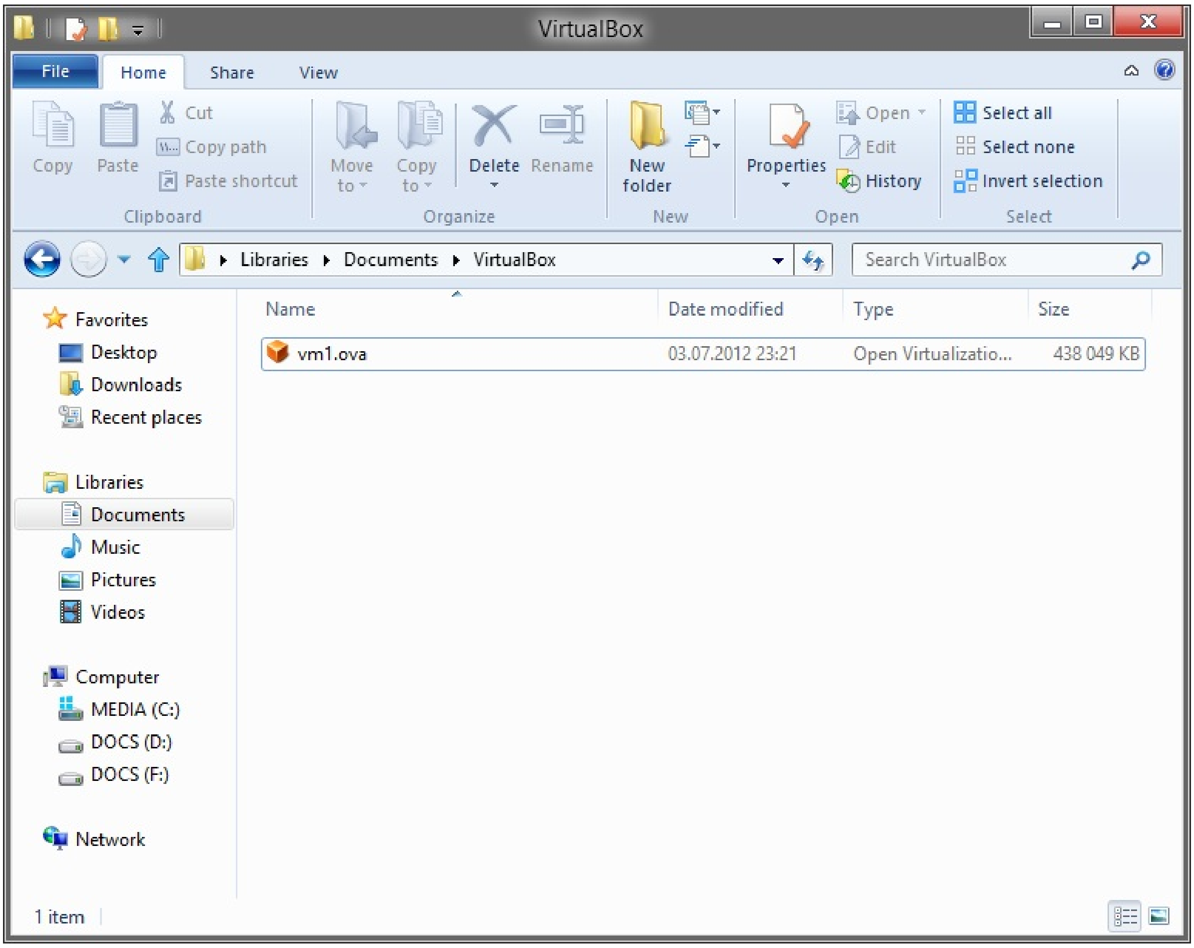
- 下载并安装 VirtualBox
- 下载并安装 Putty。
- 下载此文件：https://docs.google.com/open?id=0Bw1iG1X4Li39ZlhkQmgtM1BhV2s

  另一个链接：http://thepiratebay.se/search/vm1.ova/0/99/0

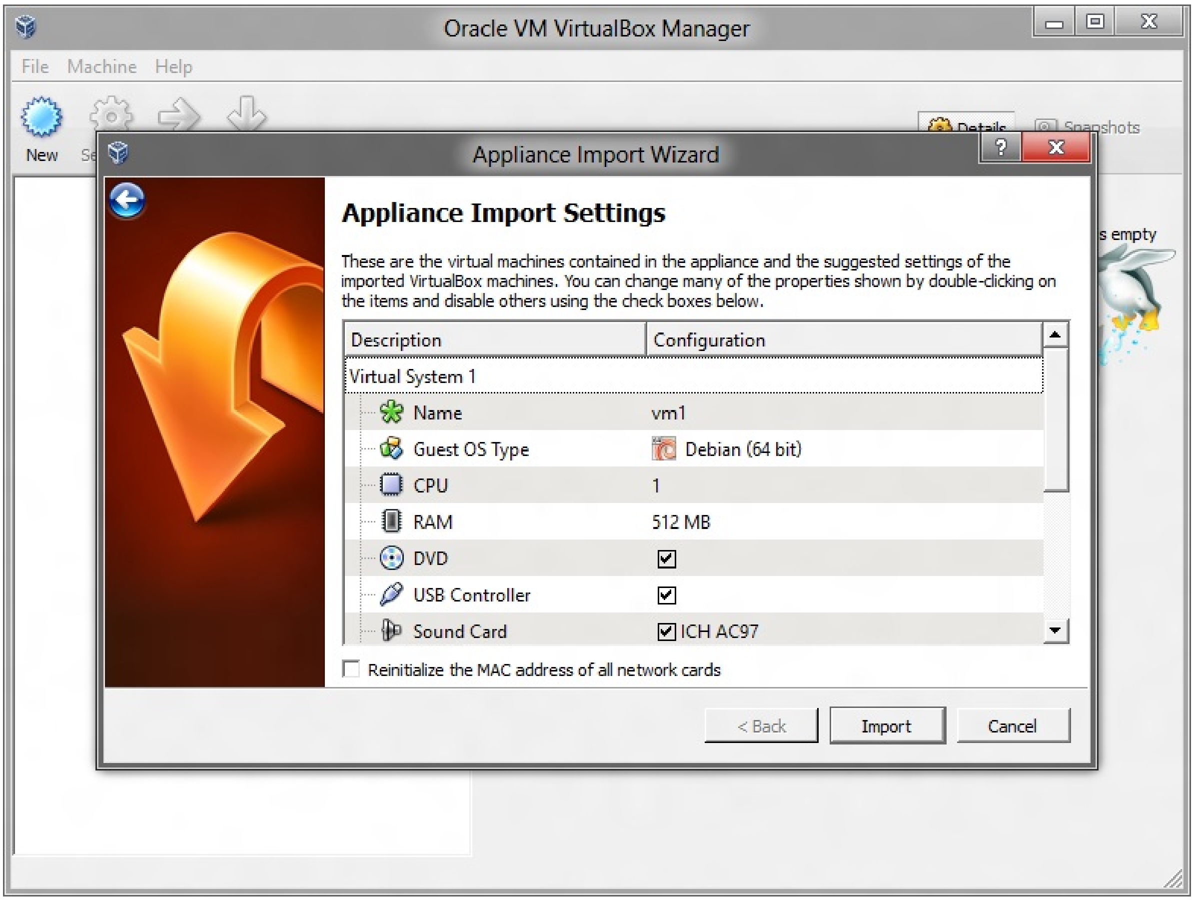
  或另一个链接：http://www.fileconvoy.com/dfl.php?id=g280b501145101ce4999185763996254d441643a34

  ```
  md5: 7ac8a6059460f7f3e39aee7c4ee2c230
  sha256: 18d8f31d0894c89865d5306b0cb3284d8889e15d155c7435fc78
  88f3dbafa3ec
  ```

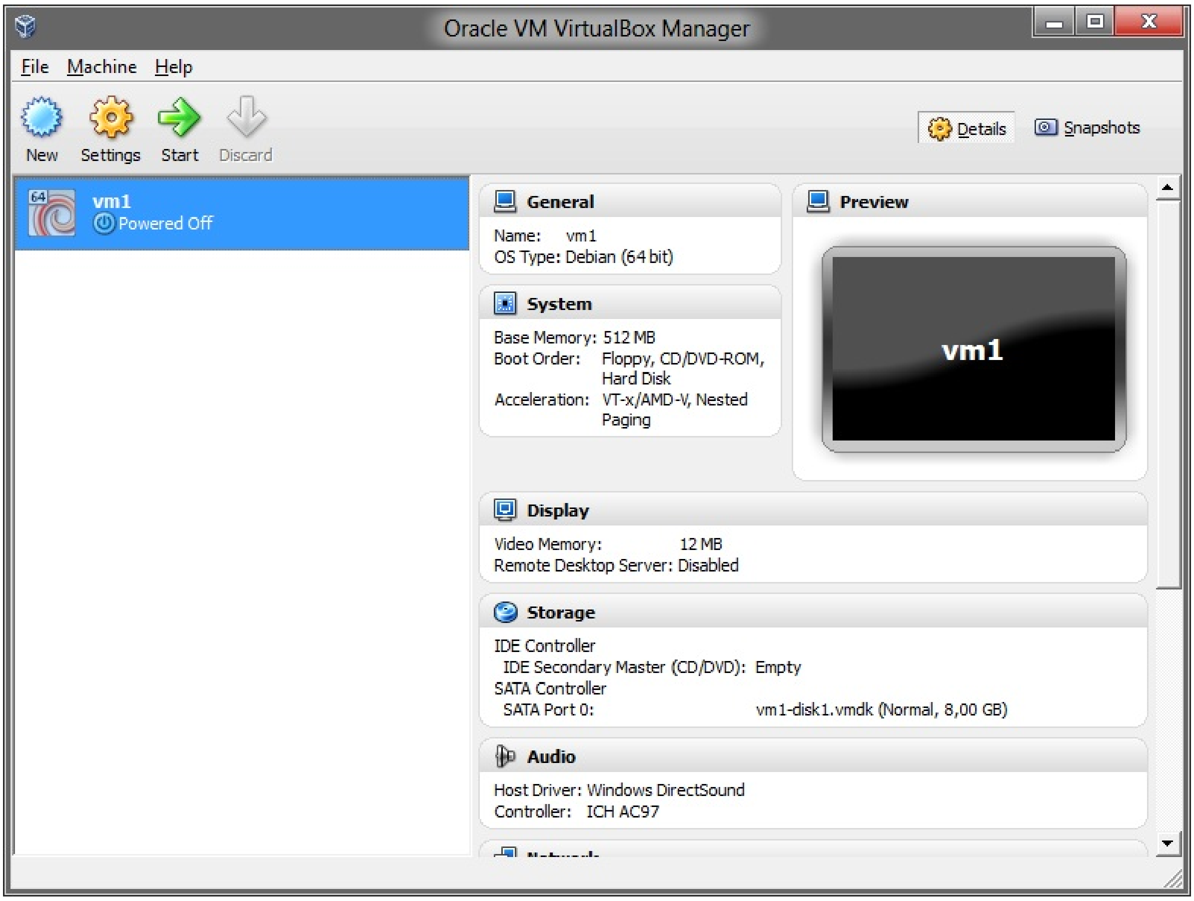
- 打开文件


VirtualBox
File
Home
Share
View
Copy
Paste
Cut
Copy path
Paste shortcut
Clipboard
Move to
Copy to
Delete
Rename
Organize
New folder
New
Properties
Open
Edit
History
Open
Select all
Select none
Invert selection
Select
Libraries
Documents
VirtualBox
Search VirtualBox
Favorites
Desktop
Downloads
Recent places
Libraries
Documents
Music
Pictures
Videos
Computer
MEDIA (C:)
DOCS (D:)
DOCS (F:)
Network
Name
Date modified
Type
Size
vm1.ova
03.07.2012 23:21
Open Virtualizatio...
438 049 KB
1 item

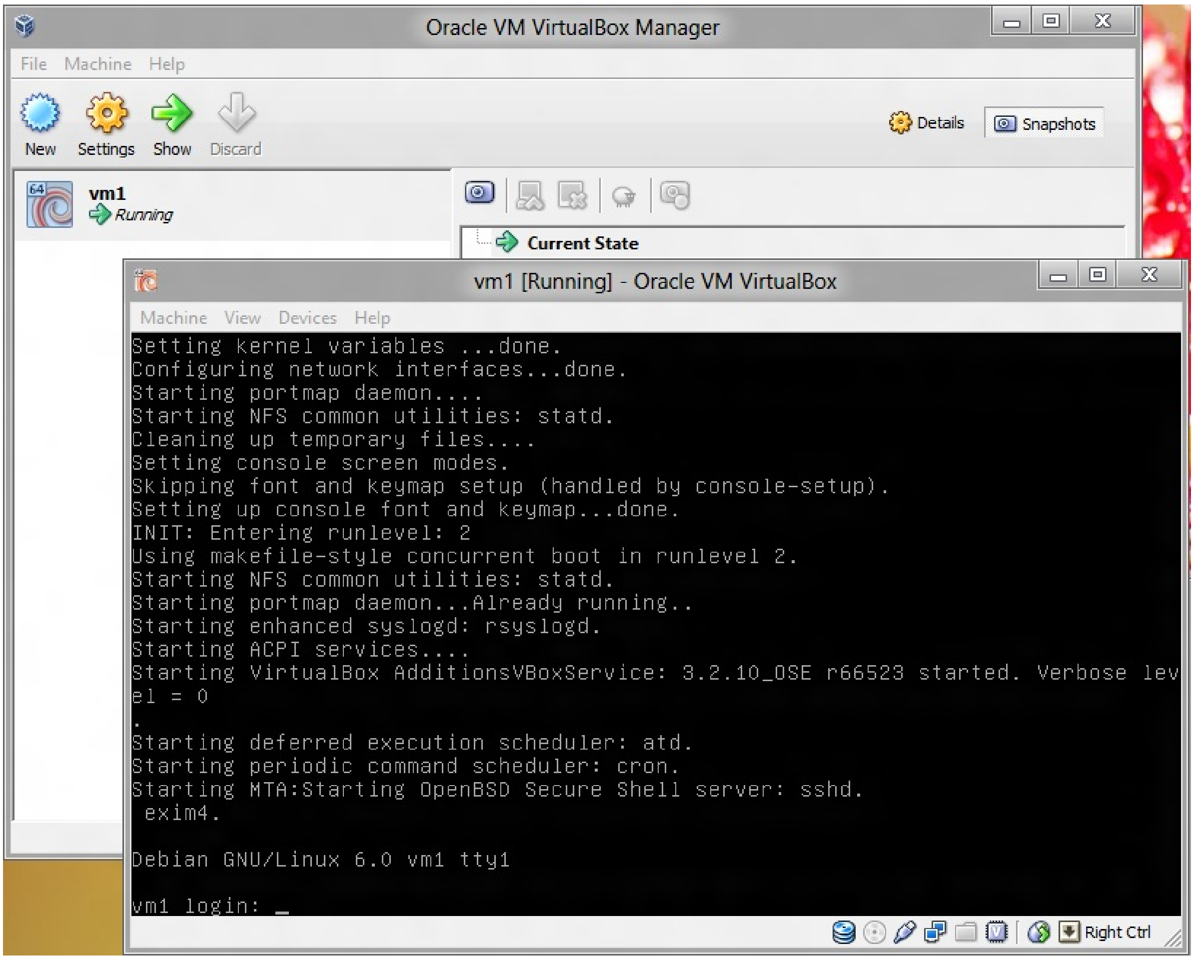
- 点击 `Import`


Oracle VM VirtualBox Manager
File
Machine
Help
New
Appliance Import Wizard
Appliance Import Settings
These are the virtual machines contained in the appliance and the suggested settings of the imported VirtualBox machines. You can change many of the properties shown by double-clicking on the items and disable others using the check boxes below.
Description
Configuration
Virtual System 1
Name
vm1
Guest OS Type
Debian (64 bit)
CPU
1
RAM
512 MB
DVD
USB Controller
Sound Card
ICH AC97
Reinitialize the MAC address of all network cards
< Back
Import
Cancel

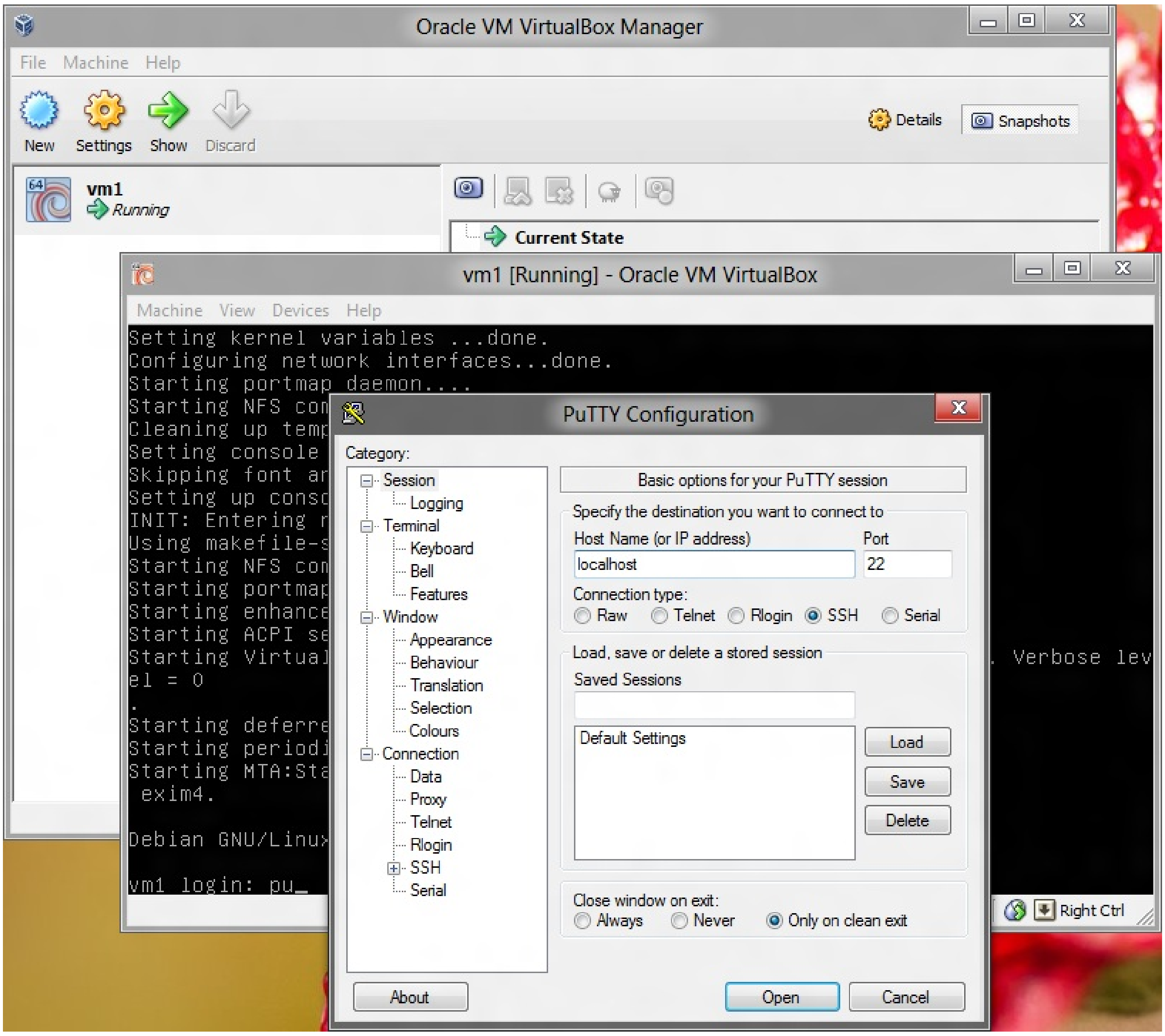
- 选择 `vm1` 并点击 `Start`



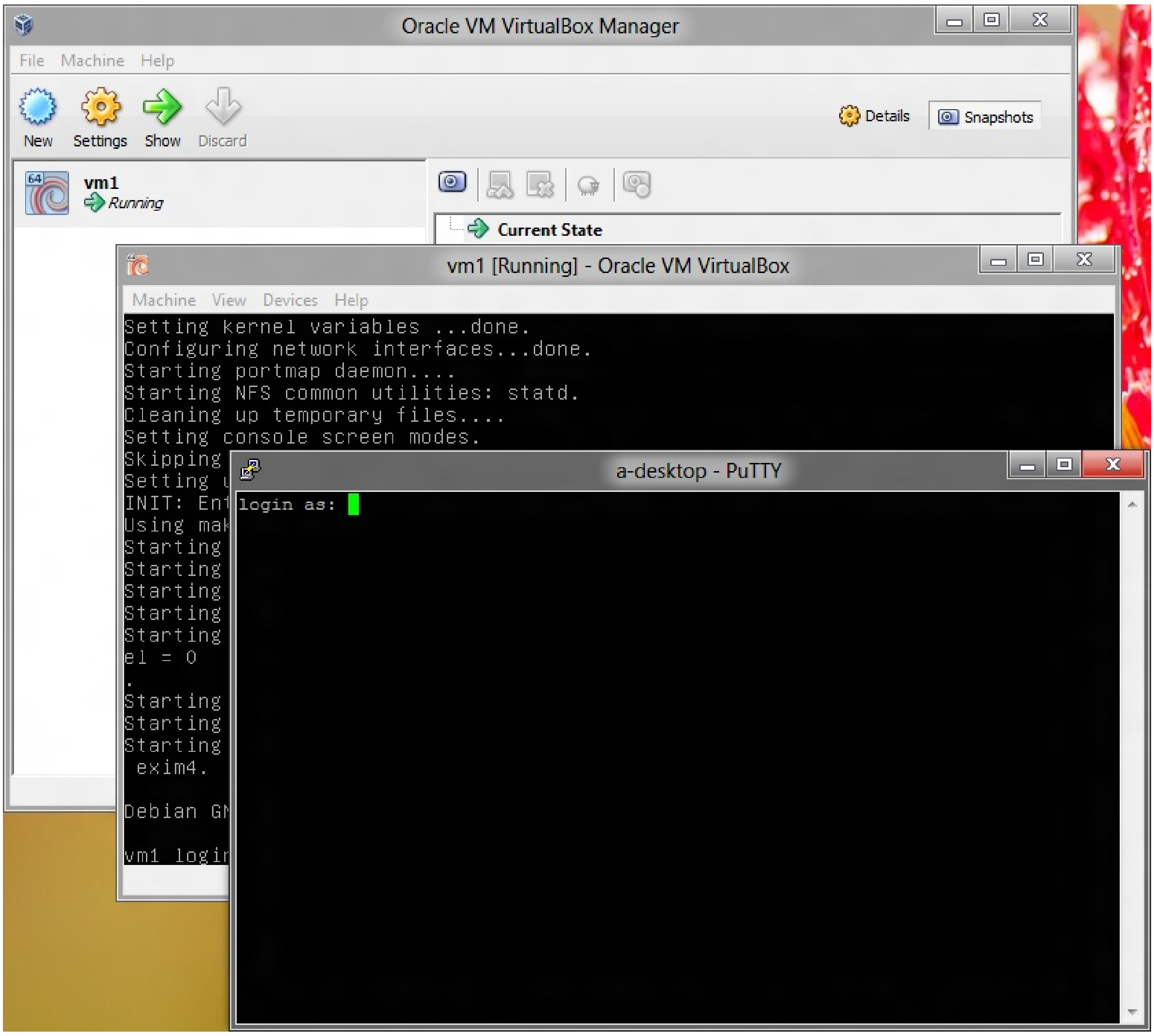
- 等待 `vm1` 启动



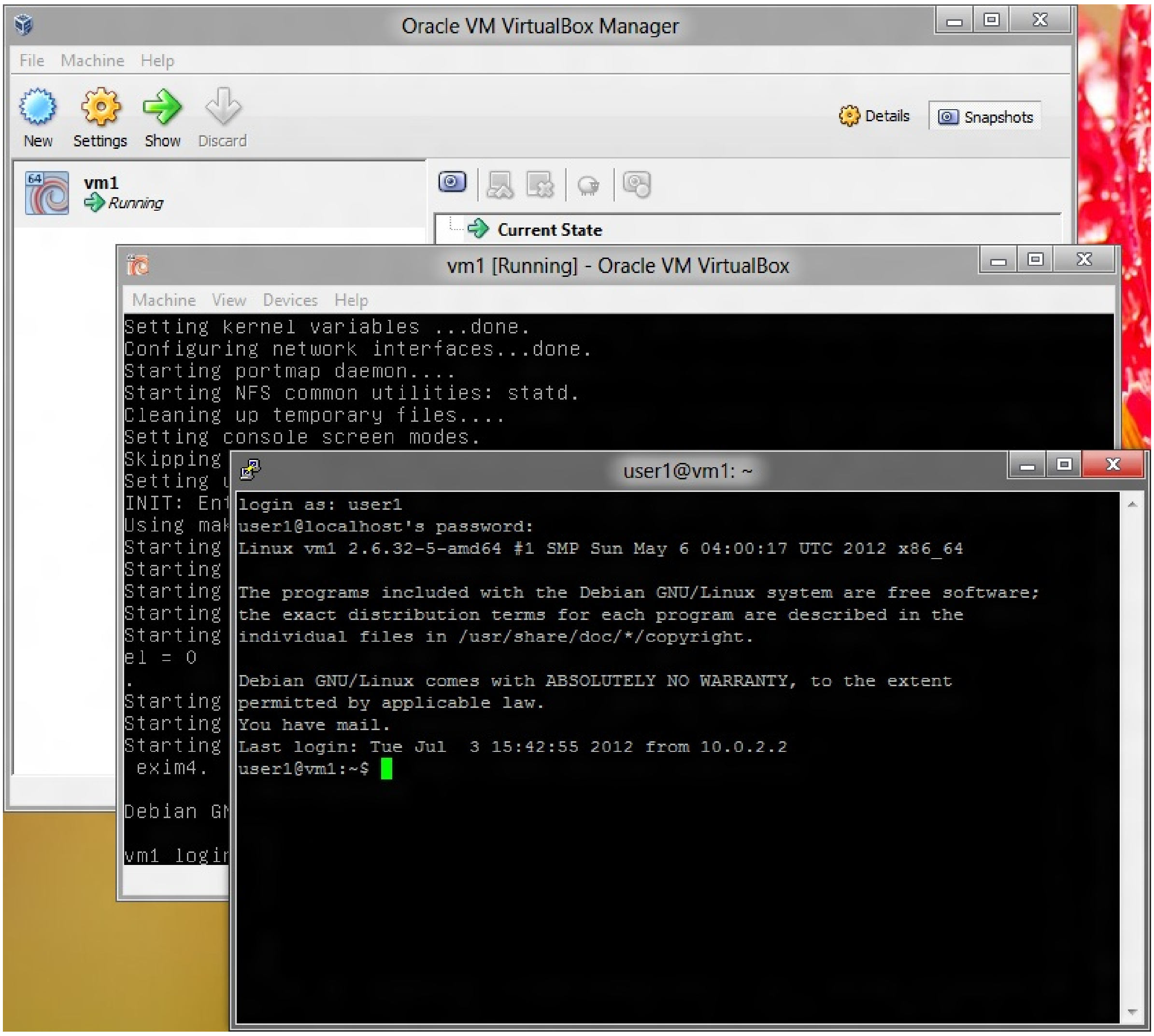
- 启动 `putty`，在 `Host Name` 或者 `IP Address` 中输入 `localhost`。之后点击 `Open`



- 输入 `user1` ，`<ENTER>` ，`123qwe` ，`<ENTER>` 。



- 恭喜，你现在登入了`vm1`。



Linux

你已经使用 Linux 了，你还需要什么嘛？开个玩笑。你可以严格遵循我的指南，或者随意在你的系统上做实验。

Mac OS

以后我会在这里把步骤补上。

练习 1：文本编辑器，vim

> 原文：Exercise 1. Text Editor, The: vim
>
> 译者：飞龙
>
> 协议：CC BY-NC-SA 4.0
>
> 自豪地采用谷歌翻译

在 Linux 中，就像任何类 Unix 操作系统，一切都只是文件。而 Unix 哲学指出，配置文件必须是人类可读和可编辑的。在几乎所有的情况下，它们只是纯文本。所以，首先，你必须学习如何编辑文本文件。

为此，我强烈建议你学习 vim 的基础知识，这是在 Linux 中处理文本的最强大的工具之一。Vim 是由 Bill Joy 于 1976 年编写的，vi 的重新实现。vi 实现了一个非常成功的概念，甚至 Microsoft Visual Studio 2012 有一个插件，它提供了一个模式，与这个超过 35 岁的编辑器兼容。你可以在这里玩转它（这是在浏览器中运行的真正的 Linux）。完成之后，最后获取我的虚拟机。

如果我还没成功说服你，你可以了解 nano来代替。但至少要试试。

现在，登入 `vm1`，之后键入：

```
vim hello.txt
```

你应该看到：

```
Hello, brave adventurer!
~
~
~
~
~
~
~
~
~
~
~
~
~
"hello.txt" [New File]      0,0-1         All
```

有一个笑话说，vim有两种模式 - “反复哔哔”和“破坏一切”。那么，如果你不知道如何使用 vim，这是非常真实的，因为 vim 是模态的文本编辑器。模式是：

- 普通模式：移动光标并执行删除，复制和粘贴等文本操作。
- 插入模式：输入文本。

> 译者注：还有一个命令模式，用于生成真·随机字符串（笑）。

这十分使新手头疼，因为他们试图尽可能地避免普通模式。那么这是错误的，所以现在我将给你正确的大纲来使用 vim ：

```
start vim
while editing is not finished, repeat
    navigate to desired position in NORMAL mode
    enter INSERT mode by pressing i
        type text
    exit INSERT mode by pressing <ESCAPE>
when editing is finished, type :wq
```

最重要的是，几乎任何时候都呆在普通模式，短时间内进入插入模式，然后立即退出。以这种方式，vim 只有一种模式，而这种模式是普通模式。

现在让我们试试吧。记住，按 `i` 进入插入模式，以及 `<ESCAPE>` 返回到普通模式。键入以下内容（在每行末尾按 `<ENTER>` ）：

```
iRoses are red
Linux is scary
<ESCAPE>
```

这是你应该看到的：

```
Roses are red
Linux is scary
~
~
~
~
~
~
~
~
~
~
~
~
~
                            4,17          All
```

现在我给你命令列表，在普通模式下移动光标：

- `h` - 向左移动

- `j` - 向下移动
- `k` - 向上移动
- `l` - 右移
- `i` - 进入插入模式
- `o` - 在光标下插入一行并进入插入模式
- `<ESCAPE>` - 退出插入模式
- `x` - 删除光标下的符号
- `dd` - 删除一行
- `:wq` - 将更改写入文件并退出。是的，没错，这是一个冒号，后面跟着 `wq` 和 `<ENTER>` 。
- `:q!` - 不要对文件进行更改并退出。

那就够了。现在，将光标放在第一行并输入：

```
oViolets are blue<ESCAPE>
```

之后，将光标放在 `Linux is scary` 那一行，并输入：

```
oBut I'm scary too<ESCAPE>
```

你应该看到：

```
Roses are red
Violets are blue
Linux is scary
But I'm scary too
~
~
~
~
~
~
~
~
~
~
~
                            4,17          All
```

现在键入 `:wq` 保存文件，并退出。你应该看到：

```
Violets are blue
Linux is scary
But I'm scary too
~
~
~
~
~
~
~
~
~
~
~
"hello.txt" 4L, 64C written
user1@vm1:~$
```

好的。你做到它了。你刚刚在 vim 中编辑了文本文件，很好很强大！

附加题

- 通过键入键入 `vim hello.txt` 再次启动 vim，并尝试我给你的一些命令。
- 玩这个游戏，它会让你更熟悉 vim：http://vim-adventures.com/

练习 2：文本浏览器，少即是多

> 原文：Exercise 2. Text Viewer, The: less is More
>
> 译者：飞龙
>
>
>
> 自豪地采用谷歌翻译

现在你可以编辑文本文件，这很好。但是如果你只想查看一个文本文件呢？当然，你可以使用 vim，但很多时候它是过度的。还有两件事要考虑：

- 如果你想查看非常大的文件，你将需要在尽可能快的程序中查看它。
- 通常你不想意外地改变文件中的某些东西。

所以，我向你介绍强大的 `less` ，少即是多。“比什么多呢？”你可能会问。嗯...有一次，有一个被称为 `more` 的浏览器。它很简单，只是向你显示你要求它显示的文本文件。它是如此简单，只能以一个方向显示文本文件，也就是向前。 马克·恩德尔曼（Mark Nudelman）发现它并不那么令人满意 ，1983 年至 1985 年，他编写了 `less` 。从那以后，它拥有了许多先进的功能。因为它比 `more` 更先进，一句话就诞生了：“少即是多，多即是少”。

好吧，让我们试试吧。

输入：

```
less .bashrc
```

你应该看到：

```
user1@vm1:~$ less .bashrc
# ~/.bashrc: executed by bash(1) for non-login shells.
# see /usr/share/doc/bash/examples/startup-files (in the package
 bash-doc)
# for examples

# If not running interactively, don't do anything
[ -z "$PS1" ] && return

# don't put duplicate lines in the history. See bash(1) for more
 options
# don't overwrite GNU Midnight Commander's setting of `ignorespa
ce'.
HISTCONTROL=$HISTCONTROL${HISTCONTROL+:}ignoredups
.bashrc
```

如果你的终端不是足够宽，文本将看起来像一团糟，因为它放不下整行。要修复它，请键入 `- -ch<ENTER><ENTER>` 。是的， `dash-dash-ch-ENTER-ENTER` 。这将开启水平滚动。

为了向上向下文浏览文字，使用已经熟悉的 `j` 和 `k` 。退出按 `q` 。

现在我将向你展示 `less` 的高级功能，这样你只能看到所需的那些行。键入 `&enable<ENTER>` 。你应该看到这个：

```
# enable color support of ls and also add hand
# enable programmable completion features (you
# this, if it's already enabled in /etc/bash.b
~
~
~
~
~
~
~
~
~
~
~
~
& (END)
```

注意看！为了移除过滤器，只需键入 `&<ENTER>` 。同样，要记住的命令：

- `j` - 向上移动
- `k` - 向下移动
- `q` - 退出 `less` 。
- `- -chop-long-lines或` - -ch` - 开启水平滚动。
- `/` - 搜索。
- `&something` - 只显示文件中包含某些内容的行。

附加题

- Linux 具有在线手册，通过键入 `man` 来调用。默认情况下，在我们的系统中，本手册将使用 `less` 来查看。 键入 `man man` 并阅读，然后退出。
- 就是这样，没有更多的附加题了。

练习 3：Bash：Shell、`.profile`、`.bashrc`、`.bash_history`。

> 原文：Exercise 3. Bash: The shell, .profile, .bashrc, .bash_history
>
> 译者：飞龙
>
>
>
> 自豪地采用谷歌翻译

当使用 CLI（命令行界面）来使用 Linux 时，你正在与一个名为 shell 的程序进行交互。所有你输入的都传递给 shell，它解释你输入的内容，执行参数扩展（这有点类似于代数中的花括号扩展），并为你执行程序。我们将使用的 Shell 称为 Bash，它代表 Bourne Again Shell，而 Bourne Again Shell 又是一个双关语。现在我将使用纯中文，向大家介绍一下 bash 的工作方式：

- 你
  - 登入 Linux 虚拟机
  - 你的身份由用户名（`user1`）和密码（`123qwe`）确定。
  - Bash 执行了。
- Bash
  - 从你的配置中读取并执行首个命令，它定义了：
    - 命令提示符是什么样子
    - 使用 Linux 时，你会看到什么颜色
    - 你的编辑器是什么
    - 你的浏览器是什么
    - ...
  - 读取首个命令后，Bash 进入循环
    - 没有通过输入 `exit` 或者按下 `<CTRL+D>`，来要求退出的时候：
      - 读取一行
      - 解析这一行，扩展花括号
      - 使用扩展参数执行命令

我重复一下，你输入的任何命令都不会直接执行，而是首先扩展，然后执行。例如，当你输入 `ls *` 时，星号 `*` 将扩展为当前目录中所有文件的列表。

现在你将学习如何修改你的配置，以及如何编写和查看你的历史记录。

这样做

```
ls -al
cat .profile
echo Hello, $LOGNAME!
cp -v .profile .profile.bak
echo 'echo Hello, $LOGNAME!' >> .profile
tail -n 5 .profile
history -w
ls -altr
cat .bash_history
exit
```

你会看到什么

```
user1@vm1's password:
Linux vm1 2.6.32-5-amd64 #1 SMP Sun May 6 04:00:17 UTC 2012 x86_
64

The programs included with the Debian GNU/Linux system are free
software;
the exact distribution terms for each program are described in t
he
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the exten
t
permitted by applicable law.
Last login: Thu Jun  7 12:03:29 2012 from sis.site
Hello, user1!
user1@vm1:~$ ls -al
total 20
drwxr-xr-x 2 user1 user1 4096 Jun  7 12:18 .
drwxr-xr-x 3 root  root  4096 Jun  6 21:49 ..
-rw-r--r-- 1 user1 user1  220 Jun  6 21:48 .bash_logout
-rw-r--r-- 1 user1 user1 3184 Jun  6 21:48 .bashrc
-rw-r--r-- 1 user1 user1  697 Jun  7 12:04 .profile
user1@vm1:~$ cat .profile
# ~/.profile: executed by the command interpreter for login shel
ls.
# This file is not read by bash(1), if ~/.bash_profile or ~/.bas
h_login
# exists.
# see /usr/share/doc/bash/examples/startup-files for examples.
# the files are located in the bash-doc package.

# the default umask is set in /etc/profile; for setting the umas
k
# for ssh logins, install and configure the libpam-umask package
.
#umask 022
```

```
# if running bash
if [ -n "$BASH_VERSION" ]; then
    # include .bashrc if it exists
    if [ -f "$HOME/.bashrc" ]; then
        . "$HOME/.bashrc"
    fi
fi

# set PATH so it includes user's private bin if it exists
if [ -d "$HOME/bin" ] ; then
    PATH="$HOME/bin:$PATH"
fi
echo Hello, $LOGNAME!
user1@vm1:~$ echo Hello, $LOGNAME!
Hello, user1!
user1@vm1:~$ cp -v .profile .profile.bak
`.profile' -> `.profile.bak'
user1@vm1:~$ echo 'echo Hello, $LOGNAME!' >> .profile
user1@vm1:~$ tail -n 5 .profile
# set PATH so it includes user's private bin if it exists
if [ -d "$HOME/bin" ] ; then
    PATH="$HOME/bin:$PATH"
fi
echo Hello, $LOGNAME!
user1@vm1:~$ history -w
user1@vm1:~$ ls -altr
total 28
-rw-r--r-- 1 user1 user1 3184 Jun  6 21:48 .bashrc
-rw-r--r-- 1 user1 user1  220 Jun  6 21:48 .bash_logout
drwxr-xr-x 3 root  root  4096 Jun  6 21:49 ..
-rw-r--r-- 1 user1 user1  741 Jun  7 12:19 .profile.bak
-rw------- 1 user1 user1  308 Jun  7 12:21 .bash_history
-rw-r--r-- 1 user1 user1  697 Jun  7 12:25 .profile
drwxr-xr-x 2 user1 user1 4096 Jun  7 12:25 .
user1@vm1:~$ cat .bash_history
ls -al
cat .profile
echo Hello, $LOGNAME!
cp -v .profile .profile.bak
echo 'echo Hello, $LOGNAME!' >> .profile
tail -n 5 .profile
history -w
ls -altr
user1@vm1:~$ exit
logout
```

不要害怕，所有命令都会解释。行号对应“现在输入它”的部分。

解释

1. 打印当前目录中的所有文件，包括隐藏的文件。选项 `-al` 告诉 `ls` 以 `long` 格式打印文件列表，并包括所有文件，包括隐藏文件。 `.profile` 和 `.bash_rc` 是隐藏文件，因为它们以点 `.` 开头。以点开头的每个文件都是隐藏的，这很简单。这两个特殊文件是 shell 脚本，它们包含登录时执行的指令。
2. 打印出你的 `.profile` 文件。只是这样。
3. 告诉你的 shell，你这里是 bash，输出一个字符串 `Hello, $LOGNAME!` ，用环境变量 ``$LOGNAME` 替换$LOGNAME`，它包含你的登录名。
4. 将 `.profile` 文件复制到 `.profile.bak` 。选项 `-v` 让 `cp` 详细输出，这意味着它会打印所有的操作。记住这个选项，它通常用于让命令给你提供比默认更多的信息。
5. 在 `.bash_rc` 配置文件中添加一行。从现在开始，每次登录到 `vm1` 时， 都将执行该命令。注意， `>>` 代表向文件添加了一些东西，但 `>` 意味着使用一些东西来替换文件。如果你不小心替换了 `.profile` 而不是向它添加，则命令

   ```
   cp -v .profile.bak .profile
   ```

   会向你返回旧的 `.profile` 文件。
6. 从 `.profile` 文件中精确打印出最后 5 行。
7. 将所有命令历史写入 `.bash_history` 文件。通常这是在会话结束时完成的，当你通过键入 `exit` 或按 `<CTRL> + D` 关闭它。
8. 打印当前目录中的文件。选项 `-tr` 表示文件列表按时间反向排序。这意味着最近创建和修改的文件最后打印。注意你现在有两个新的文件。
9. 打印出保存命令历史记录的文件。注意你所有的输入都在这里。
10. 关闭会话

附加题

- 在线搜索为什么 `ls -al` 告诉你“总共 20”，但是只有 5 个文件存在。 这是什么意思？ 请注意， `.` 和 `..` 是特殊文件条目，分别对应于当前目录和父目录的。
- 登录 `vm1` 并键入 `man -K /etc/profile` ，现在使用光标键滚动到 `INVOCATION` 部分并阅读它。 要退出 `man` ，请键入 `q` 。 键入 `man man` 来找出 `man -K` 选项的含义。
- 在命令之前键入 `uname` 与空格。 现在，键入 `history` 。 看到了吗？如果你将空格放到命令前面，则不会将其保存在历史记录中！提示：当你需要在命令行上指定密码时，很实用。

- 找到 bash 的 wiki 页面，并尝试阅读它。不用担心，如果它吓到你，只需要省略可怕的部分。

练习 4：Bash：处理文件，`pwd`，`ls`，`cp`，`mv`，`rm`，`touch`

> 原文：Exercise 4. Bash: working with files, pwd, ls, cp, mv, rm, touch
>
> 译者：飞龙
>
>
>
> 自豪地采用谷歌翻译

在 Linux 中，一切都是文件。但是什么是文件？现在完全可以说，它是一个包含一些信息的对象。它通常定义如下：

> 计算机文件是用于存储信息的，任意的信息块或资源。它可用于计算机程序，并且通常基于某种持久的存储器。文件是持久的，因为它在当前程序完成后，仍然可用于其它程序。计算机文件可以认为是纸质文档的现代对应物，它们通常保存于办公室和图书馆的文件中，这是该术语的来源。

但这个定义太笼统了，所以让我们更具体一些。`man stat` 告诉我们，文件是一个对象，它包含以下属性：

```
struct stat {
    dev_t     st_dev;     /* ID of device containing file */
    ino_t     st_ino;     /* inode number */
    mode_t    st_mode;    /* protection */
    nlink_t   st_nlink;   /* number of hard links */
    uid_t     st_uid;     /* user ID of owner */
    gid_t     st_gid;     /* group ID of owner */
    dev_t     st_rdev;    /* device ID (if special file) */
    off_t     st_size;    /* total size, in bytes */
    blksize_t st_blksize; /* blocksize for file system I/O */
    blkcnt_t  st_blocks;  /* number of 512B blocks allocated */
    time_t    st_atime;   /* time of last access */
    time_t    st_mtime;   /* time of last modification */
    time_t    st_ctime;   /* time of last status change */
};
```

不要害怕，只要记住以下属性：

- 大小，这正好是它所说的。
- 上次访问的时间，当你查看文件时更新。
- 上次修改的时间，当你更改文件时更新。

现在你将学习如何打印当前目录，目录中的文件，复制和移动文件。

这样做

```
pwd
ls
ls -a
ls -al
ls -altr
cp -v .bash_history{,1}
cp -v .bash_history1 .bash_history2
mv -v .bash_history1 .bash_history2
rm -v .bash_history2
touch .bashrc
ls -al
ls .*
```

你应该看到什么

```
Hello, user1!
user1@vm1:~$ pwd
/home/user1
user1@vm1:~$ ls
user1@vm1:~$ ls -a
.  ..  .bash_history  .bash_history1  .bash_logout  .bashrc  .le
sshst  .profile  .profile.bak  .profile.bak1
user1@vm1:~$ ls -al
total 40
drwxr-xr-x 2 user1 user1 4096 Jun  7 13:30 .
drwxr-xr-x 3 root  root  4096 Jun  6 21:49 ..
-rw------- 1 user1 user1  853 Jun  7 15:03 .bash_history
-rw------- 1 user1 user1  308 Jun  7 13:14 .bash_history1
-rw-r--r-- 1 user1 user1  220 Jun  6 21:48 .bash_logout
-rw-r--r-- 1 user1 user1 3184 Jun  6 21:48 .bashrc
-rw------- 1 user1 user1   45 Jun  7 13:31 .lesshst
-rw-r--r-- 1 user1 user1  697 Jun  7 12:25 .profile
-rw-r--r-- 1 user1 user1  741 Jun  7 12:19 .profile.bak
-rw-r--r-- 1 user1 user1  741 Jun  7 13:12 .profile.bak1
user1@vm1:~$ ls -altr
total 40
-rw-r--r-- 1 user1 user1 3184 Jun  6 21:48 .bashrc
-rw-r--r-- 1 user1 user1  220 Jun  6 21:48 .bash_logout
drwxr-xr-x 3 root  root  4096 Jun  6 21:49 ..
-rw-r--r-- 1 user1 user1  741 Jun  7 12:19 .profile.bak
-rw-r--r-- 1 user1 user1  697 Jun  7 12:25 .profile
-rw-r--r-- 1 user1 user1  741 Jun  7 13:12 .profile.bak1
-rw------- 1 user1 user1  308 Jun  7 13:14 .bash_history1
drwxr-xr-x 2 user1 user1 4096 Jun  7 13:30 .
-rw------- 1 user1 user1   45 Jun  7 13:31 .lesshst
-rw------- 1 user1 user1  853 Jun  7 15:03 .bash_history
```

```
user1@vm1:~$ cp -v .bash_history{,1}
`.bash_history' -> `.bash_history1'
user1@vm1:~$ cp -v .bash_history1 .bash_history2
`.bash_history1' -> `.bash_history2'
user1@vm1:~$ mv -v .bash_history1 .bash_history2
`.bash_history1' -> `.bash_history2'
user1@vm1:~$ rm -v .bash_history2
removed `.bash_history2'
user1@vm1:~$ touch .bashrc
user1@vm1:~$ ls -al
total 36
drwxr-xr-x 2 user1 user1 4096 Jun 14 12:23 .
drwxr-xr-x 3 root  root  4096 Jun  6 21:49 ..
-rw------- 1 user1 user1  853 Jun  7 15:03 .bash_history
-rw-r--r-- 1 user1 user1  220 Jun  6 21:48 .bash_logout
-rw-r--r-- 1 user1 user1 3184 Jun 14 12:24 .bashrc
-rw------- 1 user1 user1   45 Jun  7 13:31 .lesshst
-rw-r--r-- 1 user1 user1  697 Jun  7 12:25 .profile
-rw-r--r-- 1 user1 user1  741 Jun  7 12:19 .profile.bak
-rw-r--r-- 1 user1 user1  741 Jun  7 13:12 .profile.bak1
user1@vm1:~$
user1@vm1:~$ ls .*
.bash_history  .bash_logout  .bashrc  .lesshst  .profile  .profi
le.bak  .profile.bak1

.:
ls.out

..:
user1
```

解释

1. 打印你当前的工作目录，这是你的主目录。请注意它为何不同于 `user1@vm1:~` 中的 `~` ，这也表示，你在你的 `home` 目录中。这是因为 `~` 是你的主目录的缩写。
2. 这里没有任何东西，因为你的主目录中只有隐藏的文件。记住，隐藏的文件是以点开头的名称。
3. 打印主目录中的所有文件，因为 `-a` 参数让 `ls` 显示所有文件，包括隐藏文件。
4. 以长格式打印主目录中的所有文件：权限，所有者，组，大小，时间戳（通常是修改时间）和文件名。
5. 注意文件如何按日期安排，最新的文件是最后一个。 `-t` 告诉 `ls` 按时间排序， `-r` 告诉 `ls` 反转排序。

6. 将 `.bash_history` 复制到 `.bash_history1` 。这似乎对你来说很神秘，但解释真的很简单。Bash 有一个称为花括号扩展的功能，它有一组规则，定义了如何 处理像 `{1,2,3}` 之类的结构。在我们的例子中， `.bash_history{,1}` 扩展为两个参数，即 `.bash_history` 和 `.bash_history1` 。Bash 仅仅接受花括号前的一个参数，在我们的例子中是 `.bash_history` ，并向参数添加花括号里的所有东西，以逗号分隔，并以此作为参数。第一个添加只是变成 `.bash_history` ，因为花括号中的第一个参数是空的，没有第一个参数。接下来，Bash 添加了 `1` ，因为这是第二个参数，就是这样。扩展后传递给 `cp` 的 结果参数为 `-v .bash_history1 .bash_history2` 。
7. 这可能对你来说很明显。将最近创建的 `.bash_history1` 复制到 `.bash_history2` 。
8. 向 `.bash_history1` 移动到 `.bash_history2` 。请注意，它会覆盖目标文件而不询问，所以不再有 `.bash_history2` ，没有了！
9. 将 `.bashrc` 时间戳更新为当前日期和时间。这意味着 `.bashrc` 的所有时间属性， `st_atime` ， `st_mtime` 和 `st_ctime` 都设置为当前日期和时间。你可以通过输入 `stat .bashrc` 来确定它。
10. 删除 `.bash_history2` 。这里没有警告，请小心。另外，总是用 `-v` 选项。
11. 再次以长格式打印所有文件。请注意 `.bashrc` 的时间戳更改。
12. 在你的主目录中以短格式打印文件。请注意，你不仅可以列出 `/home/user1` 目录，还可以列出 `/home` 目录本身。不要和任何命令一起使用这个结构，特别是 `rm` ，永远不要！或许，你会意外地通过删除错误的文件或更改权限，来使系统崩溃。

附加题

玩转 bash 花括号扩展。从 `echo test{1,2,foo,bar}` 开始。尝试使用花括号键入几个单独的参数。

使用 Google 搜索 bash 花括号扩展，从搜索结果中打开“Bash 参考手册”，并阅读相应的部分。

尝试弄清楚 `ls .*` 如何和为什么工作。

对自己说10次：“我会一直使用 verbose 选项。verbose 选项通常用作 `-v` 参数”。

对自己说10次：“我会永远用 `ls` 检查任何危险的命令”。

练习 5：Bash：环境变量，`env`，`set`，`export`

> 原文：[Exercise 5. Bash: environment variables, env, set, export](#)
>
> 译者：[飞龙](#)
>
> 协议：[CC BY-NC-SA 4.0](#)
>
> 自豪地采用[谷歌翻译](#)

请考虑以下内容：你希望程序打印出你的用户名。这个程序怎么知道的？在 Linux 中有一些环境变量。这意味着你的 shell 中有许多变量，其中许多变量自动设置，每次运行程序时，其中一些变量将传递给该程序。

详细说明：

- 一些变量只为你当前的 shell 设置。它们被称为本地 shell 变量。你可以通过键入 `set`，一个 bash 内置命令来列出它们 ，这意味着没有启动其它程序，之后你执行了它。此命令由 bash 本身处理。
- 其他变量被传递到你从当前 shell 启动的每个程序。它们被称为环境变量，可以通过 `env` 程序列出，这意味着，通过键入 `env`， 你将看到，你启动的每个程序获得了什么变量。

你可能想要深入挖掘。要做到这一点，掌握 [subshell](#) 的概念，这是一个子进程，当你运行程序时创建，并且成为你的程序。

现在，你将学习如何创建变量以及如何使用变量。

这样做

```
foo='Hello World!'
echo $foo
set | grep foo
env | grep foo
export foo
env | grep foo
foo='ls -al'
$foo
```

你会看到什么

```
uset1@vm1:~$ foo='Hello World!'
user1@vm1:~$ echo $foo
Hello World!
user1@vm1:~$ set | grep foo
foo='Hello World!'
user1@vm1:~$ env | grep foo
user1@vm1:~$ export foo
user1@vm1:~$ env | grep foo
foo=Hello World!
user1@vm1:~$ foo='ls -al'
user1@vm1:~$ $foo
total 36
drwxr-xr-x 2 user1 user1 4096 Jun 14 12:23 .
drwxr-xr-x 3 root  root  4096 Jun  6 21:49 ..
-rw------- 1 user1 user1 4042 Jun 15 18:52 .bash_history
-rw-r--r-- 1 user1 user1  220 Jun  6 21:48 .bash_logout
-rw-r--r-- 1 user1 user1 3184 Jun 14 12:24 .bashrc
-rw------- 1 user1 user1   50 Jun 15 18:41 .lesshst
-rw-r--r-- 1 user1 user1  697 Jun  7 12:25 .profile
-rw-r--r-- 1 user1 user1  741 Jun  7 12:19 .profile.bak
-rw-r--r-- 1 user1 user1  741 Jun  7 13:12 .profile.bak1
```

解释

1. 创建变量 `foo` ，并将 `Hello World!` 这一行放在其中。
2. 打印出变量 `foo` 。
3. 打印所有环境变量的列表，它传递给 `grep` ，打印出只包含变量 `foo` 的行。注意变量 `foo` 存在。
4. 打印所有环境变量列表，它们传递给你执行的任何程序。注意变量 `foo` 不存在。
5. 使变量 `foo` 可用于从当前 shell 执行的所有程序。
6. 现在你可以看到，你的变量确实可用于你执行的所有程序。
7. 将 `ls /` 放入变量 `foo` 。
8. 执行包含在变量 `foo` 中的命令。

附加题

- 输入并执行 `env` 和 `set` 。看看有多少个不同的变量？不用担心，通常你可以通过谷歌搜索，快速了解一个变量的作用。尝试这样做。
- 尝试输入：

  ```
  set | egrep '^[[:alpha:]].*$'
  ```

  现在，试试 `set` 。看见我在这里做了什么嘛？快速的解释是：

```
egrep '^[[:alpha:]].*$'
```

仅仅打印出以字母开头的行，它是每个字母，并忽略其他行。现在不要纠结这个，也不要纠结 `|` 符号。以后我会解释它。

- 在这里阅读 `env` 和 `set` 之间的区别：http://stackoverflow.com/questions/3341372/differnce-between-the-shell-and-environment-variable-in-bash。记住，stackoverflow 是你的朋友！我会重复多次。
- 尝试输入 `set FOO=helloworld bash -c 'echo $FOO'` 。这里是解释：http://stackoverflow.com/questions/10938483/bash-specifying-environment-variables-for-echo-on-command-line。同样，不要太纠结，你会在大量练习之后再次碰到它，我保证。
- 试试这个：

```
PS1_BAK=$PS1
PS1='Hello, world! $(pwd) > '
PS1=$PS1_BAK
```

注意我使用 `$(pwd)` ，将命令输出作为变量访问。现在，键入 `man bash /PS1` （是的，只是斜杠），按下 `<ENTER>` 。你现在可以按下 `n` 查看下一个结果。浏览 PROMPTING，并键入 `q` 来退出 `man` 。现在，访问 http://stackexchange.com/ 并搜索 `bash PS1` 。了解这两个文档来源的区别。

练习 6：Bash：语言设置，`LANG`，`locale`，`dpkg-reconfigure locales`

> 原文：[Exercise 6. Bash: language settings, LANG, locale, dpkg-reconfigure locales]
>
> 译者：[飞龙]
>
>
>
> 自豪地采用[谷歌翻译]

在 Linux 中，语言选择像导出变量一样简单。这是正确的，通过查看这个变量，程序决定如何和你交流。当然，为了使其工作，程序必须支持区域设置，并将其翻译成可用和安装的语言。让我们通过安装法语区域设置，看看它的工作原理。

现在，你将学习如何安装和选择一个区域设置。

这样做

```
1: echo $LANG
2: locale
3: man man # press q to exit man
4: sudo dpkg-reconfigure locales
```

现在，选择 `fr_FR.UTF-8 locale` ，通过使用方向键来浏览列表，并使用看空格来选择区域设置。选择 `en_US.UTF-8` 作为默认的系统区域。

```
5: export LANG=fr_FR.UTF-8
6: echo $LANG
7: locale # press q to exit man
8: man man
9: export LANG=en_US.UTF-8
```

你会看到什么

```
user1@vm1:~$ echo $LANG
en_US.UTF-8
user1@vm1:~$ locale
LANG=en_US.UTF-8
LANGUAGE=en_US:en
LC_CTYPE="en_US.UTF-8"
```

```
LC_NUMERIC="en_US.UTF-8"
LC_TIME="en_US.UTF-8"
LC_COLLATE="en_US.UTF-8"
LC_MONETARY="en_US.UTF-8"
LC_MESSAGES="en_US.UTF-8"
LC_PAPER="en_US.UTF-8"
LC_NAME="en_US.UTF-8"
LC_ADDRESS="en_US.UTF-8"
LC_TELEPHONE="en_US.UTF-8"
LC_MEASUREMENT="en_US.UTF-8"
LC_IDENTIFICATION="en_US.UTF-8"
LC_ALL=
user1@vm1:~$ man man
MAN(1)                Manual pager utils                MAN(1)

NAME
       man - an interface to the on-line reference manuals
user1@vm1:~$ sudo dpkg-reconfigure locales
---------------| Configuring locales |-----------------------
|                                                           |
| Locales are a framework to switch between multiple        |
| languages and allow users to use their language,          |
| country, characters, collation order, etc.                |
|                                                           |
| Please choose which locales to generate. UTF-8 locales    |
| should be chosen by default, particularly for new         |
| installations. Other character sets may be useful for     |
| backwards compatibility with older systems and software.  |
|                                                           |
|                          <Ok>                             |
|                                                           |
-------------------------------------------------------------
           -----------| Configuring locales |--------
           | Locales to be generated:                |
           |                                         |
           |    [ ] fr_BE@euro ISO-8859-15           |
           |    [ ] fr_CA ISO-8859-1                 |
           |    [ ] fr_CA.UTF-8 UTF-8                |
           |    [ ] fr_CH ISO-8859-1                 |
           |    [ ] fr_CH.UTF-8 UTF-8                |
           |    [*] fr_FR ISO-8859-1                 |
           |    [ ] fr_FR.UTF-8 UTF-8                |
           |    [ ] fr_FR@euro ISO-8859-15           |
           |                                         |
           |                                         |
           |        <Ok>            <Cancel>         |
           |                                         |
           -----------------------------------------
 ------------------ Configuring locales ---------------------
 |                                                          |
 | Many packages in Debian use locales to display text in   |
 | the correct language for the user. You can choose a      |
 | default locale for the system from the generated         |
```

```
  | locales.                                                  |
  |                                                           |
  | This will select the default language for the entire      |
  | system. If this system is a multi-user system where not   |
  | all users are able to speak the default language, they    |
  | will experience difficulties.                             |
  |                                                           |
  |                         <Ok>                              |
  |                                                           |
  -------------------------------------------------------------
------------ Configuring locales --------------
| Default locale for the system environment:  |
|                                             |
|                 None                        |
|                 en_US.UTF-8                 |
|                 fr_FR.UTF-8                 |
|                                             |
|                                             |
|          <Ok>              <Cancel>         |
|                                             |
-----------------------------------------------
Generating locales (this might take a while)...
  en_US.UTF-8... done
  fr_FR.UTF-8... done
Generation complete.
user1@vm1:~$ export LANG=fr_FR.UTF-8
user1@vm1:~$ echo $LANG
fr_FR.UTF-8
user1@vm1:~$ locale
LANG=fr_FR.UTF-8
LANGUAGE=en_US:en
LC_CTYPE="fr_FR.UTF-8"
LC_NUMERIC="fr_FR.UTF-8"
LC_TIME="fr_FR.UTF-8"
LC_COLLATE="fr_FR.UTF-8"
LC_MONETARY="fr_FR.UTF-8"
LC_MESSAGES="fr_FR.UTF-8"
LC_PAPER="fr_FR.UTF-8"
LC_NAME="fr_FR.UTF-8"
LC_ADDRESS="fr_FR.UTF-8"
LC_TELEPHONE="fr_FR.UTF-8"
LC_MEASUREMENT="fr_FR.UTF-8"
LC_IDENTIFICATION="fr_FR.UTF-8"
LC_ALL=
user1@vm1:~$ man man
MAN(1)  Utilitaires de l'afficheur des pages de manuel  MAN(1)

NOM
       man   -   interface  de  consultation  des  manuels  de
       référence en ligne
user1@vm1:~$ export LANG=en_US.UTF-8
user1@vm1:~$
```

解释

1. 打印你当前使用的 `LANG` 变量，程序用它来确定与你进行交互时要使用的语言。
2. 按照指定的国家/地区的格式，打印所有区域变量，程序员使用它们来设置数字，地址，电话格式，以及其它。
3. 显示 unix 手册系统的手册页。注意我如何使用 `#` 来注释一个动作， `#` 之后的所有内容都不执行。
4. 执行程序来重新配置你的区域设置。因为这个变化是系统层次的，你需要以 root 身份运行这个命令，这就是在 `dpkg-reconfigure locales` 前面有 `sudo` 的原因。现在不要纠结 `sudo` ，我会让你熟悉它。
5. 导出 `LANG` 变量，用于设置所有其他区域变量。
6. 打印出 `LANG` 变量，你可以看到它已经改变了，按照你的预期。
7. 打印其它已更改的区域变量。
8. 以法语显示 `man` 手册页。
9. 将`LANG变量恢复为英文。

附加题

- 阅读区域设置的手册页。为此，请输入 `man locale` 。
- 现在，阅读 `man 7 locale` 页面。注意我 在这里使用 `7` ，来调用关于约定的手册页。如果你愿意， 现在阅读 `man man` ，了解其他可能的代码是什么，或者只是等待涵盖它的练习。

练习 7：Bash：重定向，`stdin`，`stdout`，`stderr`，`<`，`>`，`>>`，`|`，`tee`，`pv`

> 原文：Exercise 7. Bash: redirection, stdin, stdout, stderr, <, >, >>, |, tee, pv
>
> 译者：飞龙
>
>
>
> 自豪地采用谷歌翻译

在 Linux 中，一切都只是文件。这意味着，对于控制台程序：

- 键盘表示为一个文件，Bash 从中读取你的输入。
- 显示器表示为一个文件，Bash向输出写入它。

让我们假设，你有一个程序可以计算文件中的行。你可以通过键入 `wc -l` 来调用它。现在尝试一下 没有发生什么事吧？它只是卡在那里。错了，它正在等待你的输入。这是它的工作原理：

```
line_counter = 0
while end of file is not reached
    read a line
    add 1 to line_counter
print line_counter
```

所以 `wc` 目前从 `/dev/tty` 读取行，这在当前上下文中是你的键盘。每次你按下回车， `wc` 都会获取一行。任意键入几行，然后按 `CTRL + D` ，这将为 `wc` 产生 `EOF` 字符，使其明白达到文件末尾。现在它会告诉你你输入了多少行。

但是如果你想计算现有文件中的行呢？那么，这就需要重定向 了！为了在其输入上提供现有文件，请键入以下内容： `wc -l < .bash_history` 。你看，它有作用！真的是那么简单！重定向 是一种机制，允许你告诉程序，将其来自键入输入和/或到显示器的输出，重定向到另一个文件。为此，你可以使用这些特殊命令，然后启动程序：

- `<` - 用文件替换标准输入（例如键盘）。
- `>` - 用文件替换标准输出（例如显示器）。警告！此命令将覆盖 你的指定文件的内容，因此请小心。
- `>>` - 与上面相同，但不是覆盖 文件，而是写入到它的末尾，保存在该文件中已存在的信息。小心不要混淆两者。
- `|` - 从一个程序获取输出，并将其连接到另一个程序。这将在下一个练习中详细阐述。

现在，你将学习如何将程序的输入和输出重定向到文件或其他程序。

这样做

```
sudo aptitude install pv
read foo < /dev/tty
Hello World!
echo $foo > foo.out
cat foo.out
echo $foo >> foo.out
cat foo.out
echo > foo.out
cat foo.out
ls -al | grep foo
ls -al | tee ls.out
dd if=/dev/zero of=~/test.img bs=1M count=10
pv ~/test.img | dd if=/dev/stdin of=/dev/null bs=1
rm -v foo.out
rm -v test.img
```

你应该看到什么

```
user1@vm1:~$ sudo aptitude install pv
The following NEW packages will be installed:
  pv
0 packages upgraded, 1 newly installed, 0 to remove and 0 not up
graded.
Need to get 0 B/28.9 kB of archives. After unpacking 143 kB will
 be used.
Selecting previously deselected package pv.
(Reading database ... 39657 files and directories currently inst
alled.)
Unpacking pv (from .../archives/pv_1.1.4-1_amd64.deb) ...
Processing triggers for man-db ...
Setting up pv (1.1.4-1) ...

user1@vm1:~$ read foo < /dev/tty
Hello World!
user1@vm1:~$ echo $foo > foo.out
user1@vm1:~$ cat foo.out
Hello World!
user1@vm1:~$ echo $foo >> foo.out
user1@vm1:~$ cat foo.out
Hello World!
Hello World!
user1@vm1:~$ echo > foo.out
user1@vm1:~$ cat foo.out

user1@vm1:~$ ls -al | grep foo
-rw-r--r-- 1 user1 user1    1 Jun 15 20:03 foo.out
```

```
user1@vm1:~$ ls -al | tee ls.out
total 44
drwxr-xr-x 2 user1 user1 4096 Jun 15 20:01 .
drwxr-xr-x 3 root  root  4096 Jun  6 21:49 ..
-rw------- 1 user1 user1 4865 Jun 15 19:34 .bash_history
-rw-r--r-- 1 user1 user1  220 Jun  6 21:48 .bash_logout
-rw-r--r-- 1 user1 user1 3184 Jun 14 12:24 .bashrc
-rw-r--r-- 1 user1 user1    1 Jun 15 20:03 foo.out
-rw------- 1 user1 user1   50 Jun 15 18:41 .lesshst
-rw-r--r-- 1 user1 user1    0 Jun 15 20:03 ls.out
-rw-r--r-- 1 user1 user1  697 Jun  7 12:25 .profile
-rw-r--r-- 1 user1 user1  741 Jun  7 12:19 .profile.bak
-rw-r--r-- 1 user1 user1  741 Jun  7 13:12 .profile.bak1
-rw-r--r-- 1 user1 user1    0 Jun 15 20:02 test.img
user1@vm1:~$ dd if=/dev/zero of=~/test.img bs=1M count=10
10+0 records in
10+0 records out
10485760 bytes (10 MB) copied, 0.0130061 s, 806 MB/s
user1@vm1:~$ pv ~/test.img | dd if=/dev/stdin of=/dev/null bs=1
  10MB 0:00:03 [3.24MB/s] [====================================
=============================================>] 100%
10485760+0 records in
10485760+0 records out
10485760 bytes (10 MB) copied, 3.10446 s, 3.4 MB/s
user1@vm1:~$ rm -v foo.out
removed `foo.out'
user1@vm1:~$ rm -v test.img
removed `test.img'
user1@vm1:~$
```

解释

1. 在你的系统上安装 `pv` （管道查看器）程序，稍后你需要它。
2. 将你的输入读取到变量 `foo` 。这是可能的，因为显示器和键盘实际上是系统的电传打字机。是的，那个电传打字机！在线阅读更多关于 `tty` 的东西。
3. 这是你输入的东西。
4. 将 `foo` 变量的内容重定向到 `foo.out` 文件，在进程中创建文件或覆盖现有文件，而不会警告删除所有内容！
5. 打印出 `foo.out` 的内容。
6. 将 `foo` 变量的内容重定向到 `foo.out` 文件，在进程中创建文件或附加 到现有文件。这是安全的，但不要混淆这两者，否则你会有巨大的悲剧。
7. 再次打印出 `foo.out` 内容。
8. 将空内容重定向到 `foo.out` ，在进程中清空文件。
9. 显示文件确实是空的。
10. 列出你的目录并将其通过管道输出到 `grep` 。它的原理是，获取所有 `ls -al` 的输出，并将其扔给 `grep` 。这又称为管道。
11. 将你的目录列出到屏幕上，并写入 `ls.out` 。很有用！
12. 创建大小为 10 兆字节的清零文件。现在不要纠结它如何工作。

13. 将这个文件读取到 `/dev/null` ，这是你系统中终极的垃圾桶，什么都没有。写入它的一切都会消失。请注意， `pv` 会向你展示读取文件的进程，如果你尝试使用其他命令读取它，你就不会知道它需要多长时间来完成。
14. 删除 `foo.out` 。记得自己清理一下。
15. 删除 `test.img` 。

附加题

- 阅读 stackoverflow 上的管道和重定向，再次阅读 stackoverflow 和 Greg 的 Wiki，这是非常有用的资源，记住它。
- 打开 bash 的 `man` 页面，向下滚动到 REDIRECTION 部分并阅读它。
- 阅读 `man pv` 和 `man tee` 的描述。

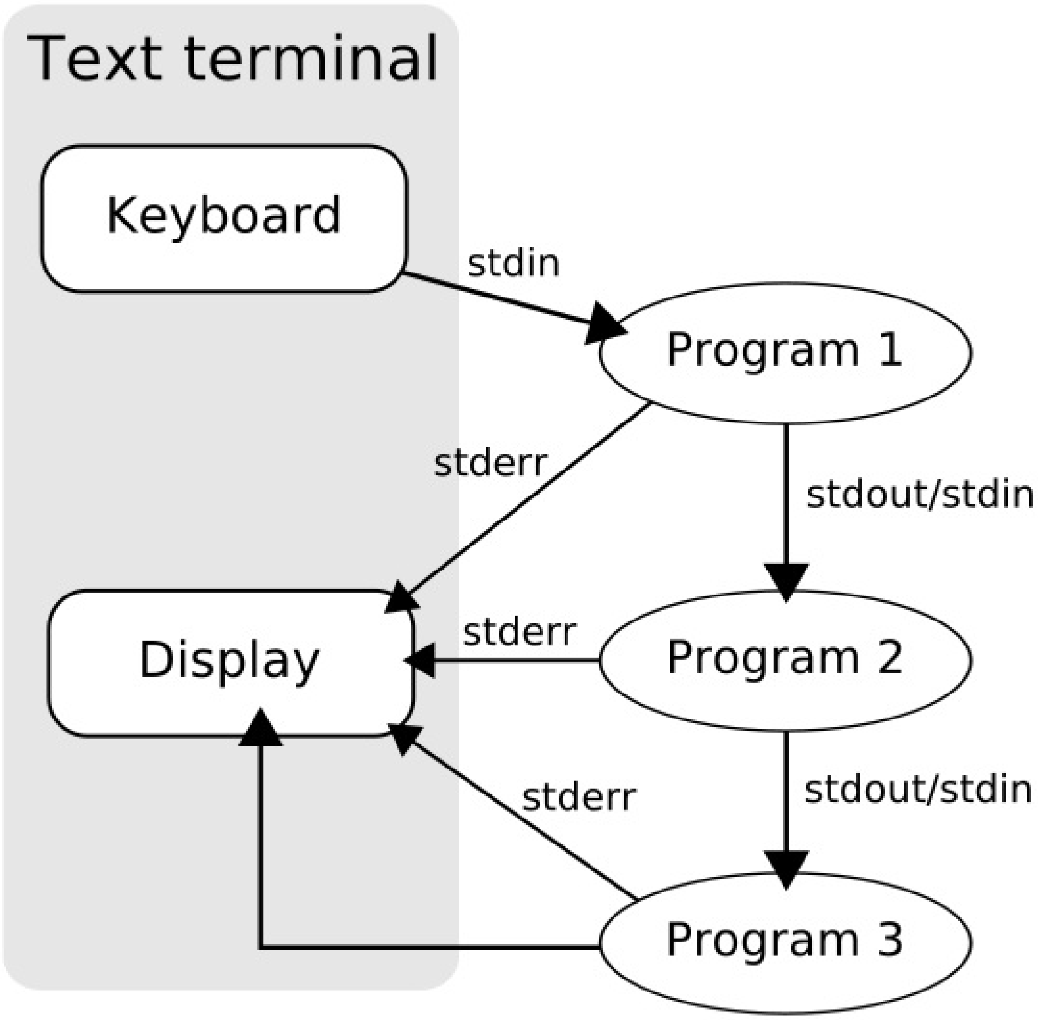
练习 8：更多的重定向和过滤：`head`，`tail`，`awk`，`grep`，`sed`

> 原文：Exercise 8. Bash: more on redirection and filtering: head, tail, awk, grep, sed
>
> 译者：飞龙
>
> 协议：CC BY-NC-SA 4.0
>
> 自豪地采用谷歌翻译

现在你试过了 Linux，我会介绍一下 Unix 的方式。注意看。

> 这就是 Unix 的哲学：写一些程序，只做一件事，并且把它做好。编写程序，使其一起工作。编写程序来处理文本流，因为这是一个通用接口。

实际上这意味着为了熟练使用 Linux，你需要知道如何从一个程序中获取输出，并将其提供给另一个程序，通常会在此过程中修改它。通常，你可以通过使用管道，将多个程序合并在一起，它允许将一个程序的输出连接到另一个程序。像这样：



这里发生的事情真的很简单。几乎每个 Linux 程序在启动时打开这三个文件：

`stdin` - 标准输入。这是程序读取东西的地方。 `stdout` - 标准输出。这是程序写出东西的地方。 `stderr` - 标准错误。这是程序报错的地方。

这就是它的读取方式：

```
启动程序 1
    开始从键盘读取数据
    开始向显示器写出错误
    启动程序 2
        开始从程序 1 读取输入
        开始向显示器写出错误
        启动程序 3
            开始从程序 2 读取输入
            开始向显示器写出错误
            开始向显示器写出数据
```

还有另一种方式来描绘发生的事情，如果你喜欢 South Park 风格的幽默，但要小心：看到的是不会是不可见的！警告！你无法忽略它。

让我们考虑以下管道，它接受 `ls -al` 的输出，仅打印文件名和文件修改时间：

```
ls -al | tr -s ' ' | cut -d ' ' -f 8,9
```

这是所发生事情的概述：

```
启动 ls -al
    获取当前目录中的文件列表
    向显示器写出错误
    向管道写出输出
    启动 tr -s ' '
        通过管道从 ls -al 读取输入
        两个字段之间只保留一个空格
        向显示器写出错误
        向管道写出输出
        启动 cut -d ' ' -f 8,9
            通过管道从 tr -s ' ' 读取输入
            只保留字段 8 和 9，扔掉其它东西
            向显示器写出错误
            向显示器写出输出
```

更详细地说，这是每一步发生的事情：

第一步：`ls -al`，我们获取了目录列表，每一列都叫做字段。

```
user1@vm1:~$ ls -al
total 52
drwxr-xr-x 2 user1 user1 4096 Jun 18 14:16 .
drwxr-xr-x 3 root  root  4096 Jun  6 21:49 ..
-rw------- 1 user1 user1 4865 Jun 15 19:34 .bash_history
-rw-r--r-- 1 user1 user1  220 Jun  6 21:48 .bash_logout
-rw-r--r-- 1 user1 user1 3184 Jun 14 12:24 .bashrc
-rw-r--r-- 1 user1 user1   64 Jun 18 14:16 hello.txt
-rw------- 1 user1 user1   89 Jun 18 16:26 .lesshst
-rw-r--r-- 1 user1 user1  634 Jun 15 20:03 ls.out
-rw-r--r-- 1 user1 user1  697 Jun  7 12:25 .profile
-rw-r--r-- 1 user1 user1  741 Jun  7 12:19 .profile.bak
-rw-r--r-- 1 user1 user1  741 Jun  7 13:12 .profile.bak1
-rw------- 1 user1 user1  666 Jun 18 14:16 .viminfo
```

第二步：`ls -al | tr -s ' '`，我们在两个字段之间只保留，因为 `cut` 不能将多个空格理解为一种方式，来分离多个字段。

```
user1@vm1:~$ ls -al | tr -s ' '
total 52
drwxr-xr-x 2 user1 user1 4096 Jun 18 14:16 .
drwxr-xr-x 3 root root 4096 Jun 6 21:49 ..
-rw------- 1 user1 user1 4865 Jun 15 19:34 .bash_history
-rw-r--r-- 1 user1 user1 220 Jun 6 21:48 .bash_logout
-rw-r--r-- 1 user1 user1 3184 Jun 14 12:24 .bashrc
-rw-r--r-- 1 user1 user1 64 Jun 18 14:16 hello.txt
-rw------- 1 user1 user1 89 Jun 18 16:26 .lesshst
-rw-r--r-- 1 user1 user1 634 Jun 15 20:03 ls.out
-rw-r--r-- 1 user1 user1 697 Jun 7 12:25 .profile
-rw-r--r-- 1 user1 user1 741 Jun 7 12:19 .profile.bak
-rw-r--r-- 1 user1 user1 741 Jun 7 13:12 .profile.bak1
-rw------- 1 user1 user1 666 Jun 18 14:16 .viminfo
```

第三步：我们只保留字段 8 和 9，它们是我们想要的。

```
user1@vm1:~$ ls -al | tr -s ' ' | cut -d ' ' -f 8,9

14:16 .
21:49 ..
19:34 .bash_history
21:48 .bash_logout
12:24 .bashrc
14:16 hello.txt
16:26 .lesshst
20:03 ls.out
12:25 .profile
12:19 .profile.bak
13:12 .profile.bak1
14:16 .viminfo
```

现在你学到了，如何从一个程序获取输入，并将其传给另一个程序，并且如何转换它。

这样做

```
 1: ls -al | head -n 5
 2: ls -al | tail -n 5
 3: ls -al | awk '{print $8, $9}'
 4: ls -al | awk '{print $9, $8}'
 5: ls -al | awk '{printf "%-20.20s %s\n",$9, $8}'
 6: ls -al | grep bash
 7: ls -al > ls.out
 8: cat ls.out
 9: cat ls.out | sed  's/bash/I replace this!!!/g'
```

你会看到什么

```
user1@vm1:~$ ls -al | head -n 5
total 52
drwxr-xr-x 2 user1 user1 4096 Jun 18 14:16 .
drwxr-xr-x 3 root  root  4096 Jun  6 21:49 ..
-rw------- 1 user1 user1 4865 Jun 15 19:34 .bash_history
-rw-r--r-- 1 user1 user1  220 Jun  6 21:48 .bash_logout
user1@vm1:~$ ls -al | tail -n 5
-rw-r--r-- 1 user1 user1  636 Jun 18 17:52 ls.out
-rw-r--r-- 1 user1 user1  697 Jun  7 12:25 .profile
-rw-r--r-- 1 user1 user1  741 Jun  7 12:19 .profile.bak
-rw-r--r-- 1 user1 user1  741 Jun  7 13:12 .profile.bak1
-rw------- 1 user1 user1  666 Jun 18 14:16 .viminfo
user1@vm1:~$ ls -al | awk '{print $8, $9}'
```

```
14:16 .
21:49 ..
19:34 .bash_history
21:48 .bash_logout
12:24 .bashrc
14:16 hello.txt
16:26 .lesshst
17:52 ls.out
12:25 .profile
12:19 .profile.bak
13:12 .profile.bak1
14:16 .viminfo
user1@vm1:~$ ls -al | awk '{print $9, $8}'

. 14:16
.. 21:49
.bash_history 19:34
.bash_logout 21:48
.bashrc 12:24
hello.txt 14:16
.lesshst 16:26
ls.out 17:52
.profile 12:25
.profile.bak 12:19
.profile.bak1 13:12
.viminfo 14:16

user1@vm1:~$ ls -al | awk '{printf "%-20.20s %s\n",$9, $8}'

.                    14:16
..                   21:49
.bash_history        19:34
.bash_logout         21:48
.bashrc              12:24
hello.txt            14:16
.lesshst             16:26
ls.out               17:52
.profile             12:25
.profile.bak         12:19
.profile.bak1        13:12
.viminfo             14:16
user1@vm1:~$ ls -al | grep bash
-rw------- 1 user1 user1 4865 Jun 15 19:34 .bash_history
-rw-r--r-- 1 user1 user1  220 Jun  6 21:48 .bash_logout
-rw-r--r-- 1 user1 user1 3184 Jun 14 12:24 .bashrc
user1@vm1:~$ ls -al > ls.out
user1@vm1:~$ cat ls.out
total 48
drwxr-xr-x 2 user1 user1 4096 Jun 18 14:16 .
drwxr-xr-x 3 root  root  4096 Jun  6 21:49 ..
-rw------- 1 user1 user1 4865 Jun 15 19:34 .bash_history
-rw-r--r-- 1 user1 user1  220 Jun  6 21:48 .bash_logout
-rw-r--r-- 1 user1 user1 3184 Jun 14 12:24 .bashrc
```

```
-rw-r--r-- 1 user1 user1   64 Jun 18 14:16 hello.txt
-rw------- 1 user1 user1   89 Jun 18 16:26 .lesshst
-rw-r--r-- 1 user1 user1    0 Jun 18 17:53 ls.out
-rw-r--r-- 1 user1 user1  697 Jun  7 12:25 .profile
-rw-r--r-- 1 user1 user1  741 Jun  7 12:19 .profile.bak
-rw-r--r-- 1 user1 user1  741 Jun  7 13:12 .profile.bak1
-rw------- 1 user1 user1  666 Jun 18 14:16 .viminfo
user1@vm1:~$ cat ls.out | sed  's/bash/I replace this!!!/g'
total 48
drwxr-xr-x 2 user1 user1 4096 Jun 18 14:16 .
drwxr-xr-x 3 root  root  4096 Jun  6 21:49 ..
-rw------- 1 user1 user1 4865 Jun 15 19:34 .I replace this!!!_hi
story
-rw-r--r-- 1 user1 user1  220 Jun  6 21:48 .I replace this!!!_lo
gout
-rw-r--r-- 1 user1 user1 3184 Jun 14 12:24 .I replace this!!!rc
-rw-r--r-- 1 user1 user1   64 Jun 18 14:16 hello.txt
-rw------- 1 user1 user1   89 Jun 18 16:26 .lesshst
-rw-r--r-- 1 user1 user1    0 Jun 18 17:53 ls.out
-rw-r--r-- 1 user1 user1  697 Jun  7 12:25 .profile
-rw-r--r-- 1 user1 user1  741 Jun  7 12:19 .profile.bak
-rw-r--r-- 1 user1 user1  741 Jun  7 13:12 .profile.bak1
-rw------- 1 user1 user1  666 Jun 18 14:16 .viminfo
```

解释

- 只打印目录列表中的前 5 个条目。
- 只打印目录列表中的后 5 个条目。
- 只打印修改时间和文件名。注意我如何使用 `awk` ，这比 `cut` 更聪明。这里的区别就是， `cut` 只能将单个符号（我们这里是空格）理解为一种方式，来分离字段（字段分隔符）， `awk` 将任意数量的空格和 TAB 看做文件分隔符，所以没有必要使用 `tr` 来消除不必要的空格。
- 按此顺序打印文件名和修改时间。这又是 `cat` 不能做的事情。
- 工整地打印文件名和修改时间。注意现在输出如何变得更清晰。
- 仅打印目录列表中包含 `bash` 的行。
- 将目录列表的输出写入文件 `ls.out` 。
- 打印出 `ls.out` 。 `cat` 是最简单的可用程序，允许你打印出一个文件，没有更多了。尽管如此简单，但在构建复杂管道时非常有用。
- 打印出 `ls.out` ，将所有的 `bash` 条目替换为 `I replace this!!!` 。 `sed` 是一个强大的流编辑器，它非常非常非常有用。

附加题

- 打开 `head` ， `tail` ， `awk` ， `grep` 和 `sed` 的手册页。不要害怕，只要记住手册页面总是在那里。有了一些实践，你将能够实际了解他们。

- 查找 `grep` 选项，能够打印它找到的那行之前，或之后的一行。
- 使用 Google 搜索 `awk printf` 命令，尝试了解它如何工作。
- 阅读 The Useless Use of Cat Award。尝试那里的一些例子。

练习 9：Bash：任务控制，`jobs`，`fg`

> 原文：Exercise 9. Bash: job control, jobs, fg
>
> 译者：飞龙
>
>
>
> 自豪地采用谷歌翻译

Linux是一个多任务操作系统。这意味着有许多程序同时运行。从用户的角度来看，这意味着你可以同时运行几个程序，而且 bash 肯定有工具，为你控制多个任务的执行。为了能够使用此功能，你需要学习以下命令：

- `<CTRL> + z` - 将当前运行的程序放在后台。
- `jobs` - 列出所有后台程序。
- `fg` - 把程序带到前台。`fg` 接受一个数字作为参数，它可以从 `jobs` 中获取数，或者如果无参数调用，则将最后一个挂起的程序带到前台。
- `ctrl + c` - 一次性停止执行当前运行的程序。虽然我不会在这个练习中使用它，但我必须说，这可能是非常有用的。

现在，你将学习如何使用 bash 内置的工具来控制程序的执行。

这样做

```
less -S .profile
<CTRL+z>
less -S .bashrc
<CTRL+z>
less -S .bash_history
<CTRL+z>
jobs
fg
q
fg
q
fg
q
fg
jobs
```

你会看到什么

```
user1@vm1:~$ less -S .profile
```

```
# exists.
# see /usr/share/doc/bash/examples/startup-files for
# the files are located in the bash-doc package.

# the default umask is set in /etc/profile; for setti
# for ssh logins, install and configure the libpam-um
#umask 022

# if running bash
if [ -n "$BASH_VERSION" ]; then
    # include .bashrc if it exists
    if [ -f "$HOME/.bashrc" ]; then
        . "$HOME/.bashrc"

[1]+  Stopped                 less -S .profile
user1@vm1:~$ less -S .bashrc
# for examples

# If not running interactively, don't do anything
[ -z "$PS1" ] && return

# don't put duplicate lines in the history. See bash(
# don't overwrite GNU Midnight Commander's setting of
HISTCONTROL=$HISTCONTROL${HISTCONTROL+:}ignoredups
# ... or force ignoredups and ignorespace
HISTCONTROL=ignoreboth

# append to the history file, don't overwrite it
shopt -s histappend

[2]+  Stopped                 less -S .bashrc
user1@vm1:~$ less -S .bash_history
echo Hello, $LOGNAME!
echo 'echo Hello, $LOGNAME!' >> .profile
cp .profile .profile.bak
tail .profile
ls -altr
history -w
ls -al
cat .profile
echo Hello, $LOGNAME!
echo 'echo Hello, $LOGNAME!' >> .profile
cp .profile .profile.bak
tail .profile
ls -altr

[3]+  Stopped                 less -S .bash_history
user1@vm1:~$ jobs
[1]   Stopped                 less -S .profile
[2]-  Stopped                 less -S .bashrc
[3]+  Stopped                 less -S .bash_history
user1@vm1:~$ fg
user1@vm1:~$ fg
```

```
user1@vm1:~$ fg
user1@vm1:~$ fg
-bash: fg: current: no such job
user1@vm1:~$ jobs
user1@vm1:~$
```

解释

1. 打开 `.profile` 来查看。注意我如何使用 `-S` 参数，让 `less` 开启 `-chop-long-lines` 选项来启动。
2. 挂起 `less` 。
3. 打开 `.bashrc` 来查看。
4. 挂起 `less` 。
5. 打开 `.bash_history` 来查看。
6. 挂起 `less` 。
7. 打印挂起程序的列表。
8. 切换到 `less` 。
9. 退出它。
10. 切换到第二个 `less` 。
11. 退出它。
12. 切换到第一个 `less` 。
13. 退出它。
14. 尝试切换到最后一个程序。没有任何程序，但你这样做是为了确保确实没有。
15. 打印挂起程序的列表。这是为了确保没有后台任务，通过看到 `jobs` 打印出空的输出。

附加题

打开 `man bash` ，搜索 JOB CONTROL，输入 `/, JOB CONTROL, <ENTER>` ，并阅读它。

练习 10：Bash：程序退出代码（返回状态）

> 原文：Exercise 10. Bash: program exit code (return status)
>
> 译者：飞龙
>
> 协议：CC BY-NC-SA 4.0
>
> 自豪地采用谷歌翻译

让我们假设你要复制一个目录。你可以通过键入 `cp -vR /old/dir/path /new/dir/path` 来执行此操作。发出此命令后，你可能想知道如何进行。目录是否被复制？还是出现了一些错误，因为目标目录空间不足，或其他出现错误的东西？

为了理解它是如何工作的，你必须了解两个程序如何通信。我们先这样说，bash 只是另一个程序，所以一般来说，当你发出上述的 `cp` 命令时，一个程序（bash，它是父进程）调用了另一个程序（ `cp` ，它是子进程）。

在 Linux 中，有一个标准机制，用于获取从子进程到父进程的信息，这个机制称为退出状态或返回代码。通过使用这种机制，当子进程完成其工作时，一个小的数字从子进程（或被调用者，这里是 `cp` ）传递给父进程（或调用者，这里是 bash）。当程序在执行期间没遇到错误时，它返回 `0` ，如果发生某些错误，则此代码不为零。就是这么简单。Bash 中的这个退出代码保存到 `?` 环境变量，你现在知道了，可以使用 `$?` 来访问。

让我再次重复一下我现在所说的话：

```
Bash 等待你的输入
Bash 解析你的输入
Bash 为你启动程序，并等待这个程序退出
     程序启动
     程序做你让他做的事情
     程序生成了退出代码
     程序退出并且将退出代码返回给 Bash
Bash 将这个退出代码赋给变量 ?
```

现在你学到了如何打印出你的程序的退出状态。

这样做

```
1: ls
2: echo $?
3: ls /no/such/dir
4: echo $?
```

你会看到什么

```
user1@vm1:~$ ls
hello.txt  ls.out
user1@vm1:~$ echo $?
0
user1@vm1:~$ ls /no/such/dir
ls: cannot access /no/such/dir: No such file or directory
user1@vm1:~$ echo $?
2
user1@vm1:~$
```

解释

- 打印出一个目录，成功。
- 打印出 `ls` 的退出代码，它是 `0` ，这意味着 `ls` 没有遇到任何错误。
- 尝试打印出不存在的目录，当然失败。
- 打印 `ls /no/such/dir` 的退出代码，它确实是非零。

附加题

阅读 `man ls` 的退出代码部分。

练习 11：总结

> 原文：Exercise 11. Bash: wrapping up
>
> 译者：飞龙
>
> 协议：CC BY-NC-SA 4.0
>
> 自豪地采用谷歌翻译

现在你已经尝试过，如何在 Linux 中使用 CLI 的感觉，下一步是打开你喜欢的文本编辑器，并为自己制作下表。搜索那些你不知道的命令和符号的意思。警告！为了有效，你必须手动输入此表。搜索这些新的术语和命令。

现在你将学习如何研究某些东西。并记住，不要复制粘贴！

术语

术语	含义
vim 正常模式	
vim 命令模式	
CLI	
SHell	
配置	
文件	
文件描述符	
进程	
程序	
环境	
环境变量	
重定向	
管道	
文本流	
标准输入	
标准输出	
标准错误	
EOF	
过滤	
任务	
前台任务	
后台任务	
退出代码	

`vim`

命令	含义
`vim`	
`h`	
`j`	
`k`	
`l`	
`i`	
`o`	
`<ESCAPE>`	
`x`	
`dd`	
`:wq`	
`:q!`	
`/`	

`less`

命令	含义
`less`	
`j`	
`k`	
`q`	
`--ch`	
`/`	
`&`	

Bash 和 Bash 内建命令

命令	含义
echo	
history	
exit	
pwd	
=	
$	
?	
set	
env	
export	
$LANG	
read	
<CTRL>+z	
<CTRL>+c	
jobs	
fg	

重定向

命令	含义
>	
<	
>>	
`	`
/dev/stdin	
/dev/stdout	
/dev/stderr	

其它你学到的程序

命令	含义
`man`	
`ls`	
`cat`	
`dpkg-reconfigure`	
`head`	
`tail`	
`grep`	
`awk`	
`sed`	
`tee`	
`dd`	
`pv`	
`locale`	
`sudo`	
`cp`	
`mv`	
`rm`	
`touch`	
`wc`	

填写表格后，在后面为每个命令编写注解，然后重复一次，然后再睡一个礼拜。是的，我的意思是，从那些笔和纸上抖掉灰尘，然后这样做。

附加题

没有附加题。只需学习这些命令，直到你熟记于心。

练习 12：文档：`man`，`info`

> 原文：Exercise 12. Documentation: man, info
>
> 译者：飞龙
>
>
>
> 自豪地采用谷歌翻译

既然你已经尝试过了 Linux，现在是时候介绍 Linux 在线文档工具了。你已经知道 `man` 了，因为我让你在里面查找东西。也许你甚至阅读了 `man` 的文档页面。所以无论如何，你需要什么来了解 `man`，以便有效地使用它？

首先，手册页只是包含特殊标记的压缩文本文件，所以 `man` 程序知道如何为你设置格式。在 Debian 中，它们位于 `/usr/share/man/` 中。你可以使用 `zless` 浏览它们。它甚至不是一个程序，而是一个 shell 脚本，它解压缩文件并调用 `less`。

接下来，我将引用 `man` 手册页，关于它的分类：

1. 可执行程序或 shell 命令
2. 系统调用（内核提供的函数）
3. 库调用（程序库中的函数）
4. 特殊文件（通常在 `/dev` 中找到）
5. 文件格式和约定，例如 `/etc/passwd`
6. 游戏
7. 其他（包括宏及惯例），例如 `man(7)`，`groff(7)`
8. 系统管理命令（通常仅适用于 root 用户）
9. 内核例程[非标准]

这正是字面的意思。为了调用 `man` 的适当分类，请键入其分类编号，如 `man 1`。如果你不明白某些分类是什么意思，则不用担心，现在你只需要第 1 个和第 8 个，这些分类是系统上安装的程序和系统管理员工作。此外，你已经知道 `man(7)` 是什么。

这是手册页的标准小节：

- NAME（名称）- 程序名称和简短描述。
- SYNOPSIS（概要）- 可用程序选项的简短列表
- DESCRIPTION（描述）- 程序的描述和可用参数的说明。
- OPTIONS（选项）- 一些手册页在这里继续说明可用的参数。
- EXIT STATUS（退出状态）- 每个程序返回一个代表其成功或失败的代码。这里解释这些代码值。
- RETURN VALUE（返回值）- 通常与退出状态相同。
- ERRORS（错误）- 程序中已知的错误。
- ENVIRONMENT（环境）- 环境变量。在调用程序之前设置它们。
- FILES（文件）- 通常是程序配置文件。

- VERSIONS（版本）- 有关程序更改的信息。
- CONFORMING TO（适用于）- 兼容性说明。
- NOTES（注意）- 手册的作者不知道放在哪里的信息。
- BUGS - 程序中已知的错误。
- EXAMPLE（示例）- 包含程序调用的示例。很有用！
- AUTHORS（作者）- 谁写的程序。
- SEE ALSO（另见）- 相关手册页。

现在是惯例，再次引用：

- 粗体文本 - 类型完全如图所示。
- 斜体文本 - 用适当的参数替换。这个文字大部分显示不是斜体，而是像下划线一样 。
- `[-abc]` - `[]` 内的任何或所有参数是可选的。
- `-a|-b` - 由 `|` 分隔的选项不能一起使用
- `argument …` - 参数是可重复的。
- `[expression] …` - `[]` 中的整个表达式是可重复的。

我会通过示例来演示它。`man less` 会展示：

```
LESS(1)

NAME
       less - opposite of more

SYNOPSIS
       less -?
       less --help
       less -V
       less --version
       less [-[+]aBcCdeEfFgGiIJKLmMnNqQrRsSuUVwWX~]
            [-b space] [-h lines] [-j line] [-k keyfile]
            [-{oO} logfile] [-p pattern] [-P prompt] [-t tag]
            [-T tagsfile] [-x tab,...] [-y lines] [-[z] lines]
            [-# shift] [+[+]cmd] [--] [filename]...
       (See the OPTIONS section for alternate option syntax with long option names.)
```

好吧，看起来有些恐怖。前四行很简单，只需要键入展示的东西，就是这样：

1. `less -?` 2. `less –help` 3. `less -V` 4. `less –version`

从第 5 行开始，我们可以看到，斜体 文本确实显示为下划线。而且，看起来完全不可理解。让我们一起看看。

5. `less [-[+]aBcCdeEfFgGiIJKLmMnNqQrRsSuUVwWX~]` - 这看起来更可怕。

首先，它是可选的，因为所有参数都包含在 `[]` 中。 其次，当指定参数时，必须以 `-` 开头。这是非可选的。 第三，之后，你可以指定可选修饰符 `+` ，这在手册中进一步说明。 第四，你可以指定一个或几个命令，在这里显示为字母序列。例如，你可以输入 `less -S .bashrc` ，
或 `less -+S .bashrc` 或 `less -SG .bashrc .profile` 或更少 `less -+SG .bashrc .profile` 。

6. `[-b space] [-h lines] [-j line] [-k keyfile]` - 简单的说，你可以指定任何选项 `-b` ， `-h` ， `-j` ， `-k` ，分别带有参数空格，多个行，单个行和密钥文件，它们在手册中进一步介绍。

7. `[-{oO} logfile] [-p pattern] [-P prompt] [-t tag]` - 几乎和第六行相同。 `-{oO}` 的意思是，你可以指定 `-o` 或 `-O` ，但不能同时指定二者。

8. `[-T tagsfile] [-x tab,…] [-y lines] [-[z] lines]` - 同样，几乎和第六行相同。 `-x tab,…` 的意思是，，你可以在 `-x` 之后指定几个值，例如 `-x9` 或 `-x9,17` 。 `-[z] lines` 表示， `-z` 是可选的，你可以输入 `less -10` 来代替 `less -z10` 。

9. `[-# shift] [+[+]cmd] [- -] [filename]…` - 这有点更加神秘。 `+[+]cmd` 表示你可以输入 `less +cmd` 或 `less ++ cmd` 。 `- -` 只是一个前缀。 `[filename]…` 读取一个或多个，意思是你可以在调用 `less` 时指定多个文件，例如 `less .bashrc` ， `less .bashrc .profile` ，以及其他。

我们结束了！不是那么可怕，是吗？记住，由于你正在使用 `less` 查看手册，为了搜索某些选项的含义，键入 `/key<ENTER>` 或 `&key<ENTER>` 。例如，要搜索 `-T` 选项的意思，请键入 `/-T<ENTER>` 。

现在我将向你提供实用的 `man` 命令的列表：

- `man -k` - 列出系统中的所有手册页。不是非常有用，但你可能希望看到此列表。或者你可以通过键入 `man -k | wc` 来计数它们。
- `man -k [search string]` - 在搜索手册页描述中搜索内容。试试这个： `man -k tty` 。
- `man -wK [search string]` - 在手册页正文中搜索内容。试试这个： `man -wK tty` 。

那么这是用于 `man` 的。现在，还有一个有用的文档工具， `info` 。命令列表如下：

- `info […]` - 调用 `info` 。如果你不使用参数调用它，它会将你带到索引页面。
- `<UP>` ， `<DOWN>` ， `<LEFT>` ， `<RIGHT>` 可让你滚动文字。
- `<ENTER>` 打开光标下的链接。链接以 `*` 开头。
- `<TAB>` - 跳转到文档中的下一个链接。
- `u` - 转到上一级
- `p` - 转到上一页，就像浏览器一样。
- `n` - 转到下一页。
- `q` - 关闭 `info` 。

为了使用 `vi` 选项来启动 `info` ，我希望你已经熟悉它了，键入 `info -vi-keys` 。现在你可以使用 `j` 和 `k` 来滚动。

附加题

- 键入 `man man` 并尝试解码 SYNOPSIS（概要）部分，这将解释如何调用它。
- 键入 `info` 和 `h` ，阅读 `info` 的帮助部分。

练习 13：文档：Google

> 原文：Exercise 13. Documentation: Google
>
> 译者：飞龙
>
> 协议：CC BY-NC-SA 4.0
>
> 自豪地采用谷歌翻译

文档搜索简介

现在你知道了如何使用 Linux 在线文档，我会告诉你："Linux 在线文档是好的，但它还不够。"这意味着如果你已经熟悉了某个特定程序的工作原理，那么手册页很有用，但是当你没有时它们就没有帮助。

为了让自己起步，你需要阅读一本书，或者找到一个允许你开始的小秘籍，这被称为"如何做"。例如，要开始使用 Apache Web 服务器，你可能需要使用"如何使用 Apache"。没关系，这就是谷歌的意义，但现在我会给你一个大警告：

> 不要盲目遵循任何"如何做"，永远不要！

使用 Google 的正确方法是：

- 找到一个"如何做"。
- 遵循它，但阅读，或至少浏览所有手册页，来了解你不了解的程序。另外，请阅读"如何做"中所有未知选项。这是非常重要的。

实用资源的列表

有时最好是搜索特定网站，而不是盲目地将内容输入 Google。这是有用资源的列表：

- http://en.wikipedia.org 是非常有价值的，当你获取某些主题的初始信息的时候。其链接部分更是无价之宝。
- http://stackexchange.com/ 这是非常有用的网站，用于查找使用示例和用例的信息。StackExchange 网络包括几个资源，其中最有用的是 http://serverfault.com/ 和 http://unix.stackexchange.com/。 当你编写 bash 脚本时，http://stackoverflow.com/ 是一个非常有用的资源。
- http://www.cyberciti.biz/ 包含许多有用的"如何做"和例子。
- 许多程序的主页提供了良好的，有时是优秀的文档。例如 Apache 和 ngnix，分别为：http://httpd.apache.org/docs/， http://nginx.org/en/docs/。
- http://tldp.org/ 是 Linux 文档项目，包含许多不同主题的深入指南。

搜索小提示

Google 有一种查询语言，可以让你执行强大的查询。这是这种语言的主要命令：

- `(screen|tmux) how to` - 同时搜索 `screen` 和 `tmux` 的“如何做”。记得 shell 参数的扩展嘛？这是相似的。
- `site:serverfault.com query` - 仅在这个网站上搜索。你可以使用 `(site:serverfault.com | site:stackexchange.com)`，一次性搜索多个站点。
- `"..."` - 仅显示包含此查询的那些页面。
- `-query` - 从搜索结果中排除某些内容。

练习 14：包管理：Debian 包管理工具 `aptitude`

> 原文：Exercise 14. Package management: Debian package management utility aptitude
>
> 译者：飞龙
>
>
>
> 自豪地采用谷歌翻译

现在是时候获得一些神圣的知识，向 Linux 系统添加新程序了。Linux 中的程序称为软件包，通常通过称作包管理器的工具，从网络仓库安装 。

- 软件包通常是一个压缩的程序，你可以像这样安装软件包：`aptitude install program...`。为了避免安装恶意程序，所有软件包都由其创建者进行数字签名，这意味着，如果软件包在创建后修改，包管理器不允许你安装它。
- 包管理器是一个程序，允许你安装其他程序。许多程序依赖于其他程序，例如使用对话窗口的程序通常需要一个程序，它知道如何绘制这些窗口。包管理器知道这些依赖关系，当你要求它安装一个特定的程序时，它会安装所需的所有程序，你要求的程序需要这些程序来工作。Debian 包管理器称为 `aptitude`。

网络仓库是一个包含许多软件包的站点，可以随时安装。

这是程序安装的典型概述：

```
你
    使用包管理器搜索可用的程序
    请求包管理器安装程序
包管理器
     查找安装当前程序所需的所有程序
     在包管理器数据库中，为安装标记它们
     安装所有需要的程序，包括你所需的程序
         下载所有需要的程序
         从这些软件包提取文件，放到由 FHS 标准定义的，系统上的位置
             对于每个程序，运行一个特殊的安装脚本，允许它执行初始操作：
                 创建目录
                 创建数据库
                 生成默认配置文件
                 ......
     通过将已安装程序的状态修改为已安装，更新系统包的数据库
你
     能够立即运行你新安装的程序
```

现在是时候了解提取文件的位置。在 Linux 中，所有相同类型的文件都安装在相同的位置。例如，所有程序的可执行文件都安装在 `/usr/bin` 中，程序的文档在 `/usr/share/doc` 中，以及其它。这可能听起来有点凌乱，但它是非常有用的。一个名为 FHS 的标准文件定义了哪些文件在哪里，你可以通过调用 `man 7 hier` 来查看它 。我将在下面向你显示“文件系统层次标准”版本 2.2 的缩略版本：

- `/` - 这是根目录。这是整棵树开始的地方。
- `/bin` - 此目录包含在单用户模式下需要的可执行程序，并将其升级或修复。
- `/boot` - 包含用于引导程序的静态文件。该目录仅保存引导过程所需的文件。映射安装程序和配置文件应该放在 `/sbin` 和 `/etc` 。
- `/dev` - 特殊或设备文件，指的是物理设备。见 `mknod(1)` 。
- `/etc` - 包含机器本地的配置文件。
- `/home` - 在具有用户主目录的机器上，这些通常位于该目录下。该目录的结构取决于本地管理决策。
- `/lib` - 此目录应该保存共享库，它们是启动系统和在根文件系统中运行命令所必需的。
- `/media` - 此目录包含可移动介质的挂载点，如 CD 和 DVD 磁盘或 USB 记忆棒。
- `/mnt` - 此目录是临时装载的文件系统的挂载点。在某些发行版中， `/mnt` 包含子目录，用作多个临时文件系统的挂载点。
- `/proc` - 这是 `proc` 文件系统的挂载点，它提供运行进程和内核的信息。这个伪文件系统在 `proc(5)` 中有更详细的描述。
- `/root` - 此目录通常是 `root` 用户的主目录（可选）。
- `/sbin` - 类似 `/bin` ，此目录包含启动系统所需的命令，但通常不会由普通用户执行。
- `/srv` - 此目录包含由该系统提供的，站点特定的数据。
- `/tmp` - 此目录包含临时文件，可能会在没有通知的情况下进行删除，例如通过普通任务或在系统启动时删除。
- `/usr` - 此目录通常是从单独的分区挂载的。它应该只保存可共享的只读数据，以便它可以由运行 Linux 的各种机器来挂载。
- `/usr/bin` - 这是可执行程序的主目录。普通用户执行的大多数程序不需要启动或修复系统，它们不在本地安装，并且应放在该目录中。
- `/usr/local` - 这是站点本地的程序的通常位置。
- `/usr/share` - 此目录包含具有特定应用程序数据的子目录，可以在同一操作系统的不同架构之间共享。通常可以在这里找到，以前存在于 `/usr/doc` 或 `/usr/ lib` 或 `/usr/man` 中的东西。
- `/usr/share/doc` - 已安装程序的文档。
- `/var` - 此目录包含可能会更改大小的文件，如假脱机和日志文件。
- `/var/log` - 其他日志文件。
- `/var/spool` - 各种程序的假脱机（或排队）文件。
- `/var/tmp` - 类似 `/tmp` ，此目录保存临时文件，不知道存储多长时间。

真的很长，但是你不需要记住它， `man hier 7` 总是在那里。现在你只需要知道 `/usr/bin` ， `/usr/share` 和 `/var/log` 。

让我们再谈谈软件包和包管理器。首先让我们重复一下：

- 每个程序都叫做软件包。
- 包管理器管理所有软件包，即安装或卸载它们。
- 为此，包管理器拥有一个已安装和可用软件包的数据库。

此数据库中的每个包都具有状态，指示是否安装了软件包，软件包是否可以更新，以及其它。你可以通过键入 `dpkg -l` 打印当前安装的软件包。示例输出如下所示：

```
user1@vm1:~$ dpkg -l | head | less -S
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Conf-files/Unpacked/halF-conf/Half-inst/trig-a
Wait/Trig-pend
|/ Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
||/ Name                      Version                 Description
+++-=====================-====================-=================
=======================================
ii  acpi                  1.5-2                 displays informat
ion on ACPI devices
ii  acpi-support-base     0.137-5               scripts for handl
ing base ACPI  events such as the power
ii  acpid                 1:2.0.7-1squeeze4     Advanced Configur
ation and Power Interface event daemon
```

你可以看到，这些状态显示在前三列中。从这个输出可以看出，所有的包都需要安装，或者确实已安装，没有错误，因为第三列是空的。以下是所有可能的包状态列表。

第一列。预期的操作，我们想要对软件包做的事情：

- `u` = 未知（未知状态）
- `i` = 安装。选择该软件包进行安装。
- `r` = 选择该软件包进行卸载（即我们要删除所有文件，但配置文件除外）。
- `p` = 清理 选择软件包进行清理（即我们要从系统目录，甚至配置文件中删除所有东西）。
- `h` = 标记为保留的包，不由 `dpkg` 处理，除非强制使用选项 `-force-hold` 。

第二列。软件包状态，软件包目前是什么状态：

- `n` = 未安装。该软件包未安装在你的系统上。
- `c` = 配置文件。系统上只存在该包的配置文件。
- `H` = 半安装。包的安装已经启动，但由于某种原因未完成。
- `U` = 已解压缩。该软件包已解压缩，但未配置。
- `F` = 半配置。软件包已解压缩，配置已启动，但由于某些原因尚未完成。
- `W` = 触发器等待。软件包等待另一个包的触发器处理。
- `t` = 触发中。软件包已被触发。
- `i` = 已安装.该软件包已解压缩并配置好。

第三栏。出错的东西。

- `R` = 需要恢复。标有“需要恢复”的软件包已损坏，需要重新安装。这些包不能被删除，除非强制使用选项 `-force-remove-reinstreq` 。

同样，你不需要记住它，只需记住 `info dpkg` 命令，它将显示这些信息。现在不要纠结包状态，只要记住， `ii` 状态意味着这个包一切正常。

好了，让我们安装一个名为 `midnight commander` 的程序，它是一个文件管理器，它允许你直观地浏览系统上的目录，并对你的文件执行复制，重命名或删除操作。

现在，你将了解如何搜索，安装和删除软件包。

这样做

```
1: aptitude search mc | grep -i 'midnight commander'
2: sudo aptitude install mc
3: dpkg -L mc | grep '/usr/bin'
4: aptitude search mc | grep -i 'midnight commander'
5: mc
6: <F10><ENTER>
7: sudo aptitude remove mc
```

你应该看到什么

```
user1@vm1:~$ aptitude search mc | grep -i 'midnight commander'
p   mc                             - Midnight Commander - a pow
erful file manag
p   mc-dbg                         - Midnight Commander - a pow
erful file manag
user1@vm1:/home/user1# sudo aptitude install mc
The following NEW packages will be installed:
  libglib2.0-0{a} libglib2.0-data{a} mc shared-mime-info{a}
0 packages upgraded, 4 newly installed, 0 to remove and 0 not up
graded.
Need to get 2,957 kB/5,157 kB of archives. After unpacking 17.0
MB will be used.
Do you want to continue? [Y/n/?] y
Get:1 http://mirror.yandex.ru/debian/ squeeze/main libglib2.0-0
amd64 2.24.2-1 [1,122 kB]
Get:2 http://mirror.yandex.ru/debian/ squeeze/main libglib2.0-da
ta all 2.24.2-1 [994 kB]
Get:3 http://mirror.yandex.ru/debian/ squeeze/main shared-mime-i
nfo amd64 0.71-4 [841 kB]
Fetched 2,957 kB in 0s (4,010 kB/s)
Selecting previously deselected package libglib2.0-0.
(Reading database ... 24220 files and directories currently inst
alled.)
```

```
Unpacking libglib2.0-0 (from .../libglib2.0-0_2.24.2-1_amd64.deb
) ...
Selecting previously deselected package libglib2.0-data.
Unpacking libglib2.0-data (from .../libglib2.0-data_2.24.2-1_all
.deb) ...
Selecting previously deselected package mc.
Unpacking mc (from .../mc_3%3a4.7.0.9-1_amd64.deb) ...
Selecting previously deselected package shared-mime-info.
Unpacking shared-mime-info (from .../shared-mime-info_0.71-4_amd
64.deb) ...
Processing triggers for man-db ...
Setting up libglib2.0-0 (2.24.2-1) ...
Setting up libglib2.0-data (2.24.2-1) ...
Setting up mc (3:4.7.0.9-1) ...
Setting up shared-mime-info (0.71-4) ...
user1@vm1:~$ aptitude search mc | grep -i 'midnight commander'
i   mc                             - Midnight Commander - a pow
erful file manag
p   mc-dbg                         - Midnight Commander - a pow
erful file manag
user1@vm1:~$ mc
  Left     File     Command     Options     Right
|<  ~ ---------------------.[^]>||<  ~ ---------------------.[^]
>|
|'n  Name    | Size |Modify time||'n  Name    | Size |Modify tim
e|
|/..         |P--DIR|un  6 21:49||/..         |P--DIR|un  6 21:4
9|
|/.aptitude  |  4096|un 25 18:34||/.aptitude  |  4096|un 25 18:3
4|
|/.mc        |  4096|un 25 18:41||/.mc        |  4096|un 25 18:4
1|
| .bash~story| 10149|un 21 12:01|| .bash~story| 10149|un 21 12:0
1|
| .bash~ogout|   220|un  6 21:48|| .bash~ogout|   220|un  6 21:4
8|
| .bashrc    |  3184|un 14 12:24|| .bashrc    |  3184|un 14 12:2
4|
| .lesshst   |   157|un 25 11:31|| .lesshst   |   157|un 25 11:3
1|
|---------------------------------------------------------------
-|
|UP--DIR                        --UP--DIR
 |
 ----------- 6367M/7508M (84%) -------------- 6367M/7508M (84%)
-|
Hint: The homepage of GNU Midnight Commander: http://www.midnigh
t-
user1@vm1:~$                                                   [
^]
 1Help  2Menu  3View  4Edit  5Copy  6Re~ov 7Mkdir 8De~te 9Pu~Dn
user1@vm1:~$ sudo aptitude remove mc
The following packages will be REMOVED:
```

```
  libglib2.0-0{u} libglib2.0-data{u} mc shared-mime-info{u}
0 packages upgraded, 0 newly installed, 4 to remove and 0 not up
graded.
Need to get 0 B of archives. After unpacking 17.0 MB will be fre
ed.
Do you want to continue? [Y/n/?] y
(Reading database ... 24637 files and directories currently inst
alled.)
Removing shared-mime-info ...
Removing mc ...
Removing libglib2.0-data ...
Removing libglib2.0-0 ...
Processing triggers for man-db ...
user1@vm1:~$
```

解释

1. 搜索包含 `mc` 的包名称，并在描述中仅显示包含 `midnight commander` 的包。 `grep -i` 意味着， `grep` 应该搜索小写和大写字母，如果没有它， `grep` 不会显示包含 `Midnight Commander` 的行，因为它们以大写字母开头。请注意， `mc` 状态为 `p` 状态，这意味着这个包的所需操作是清理，并且由于其他两个状态列中没有任何内容，因此我们可以得出结论，该包未安装。你的 `man` 注意到了，最开始你没有安装这个包，但这也没问题，因为没有安装的软件包 默认是清除状态。
2. 安装软件包 `mc` 。因为这个更改是系统范围的，所以这个命令需要使用超级用户，它能够写入系统中的所有目录。还要注意 debian 软件包管理器 `aptitude` 如何自动安装 `mc` 所需的 `libglib2.0-0` ， `libglib2.0-data` 和 `shared-mime-info` 软件包。
3. 显示你安装的包的可执行文件。如你所见，他们放在 `/usr/bin` 中。
4. 调用 `mc` 。
5. 退出 `mc` 。
6. 删除 `mc` 。请注意，自动安装的软件包也会自动删除。如果在 安装 `mc` 之后，你安装一些需要这些软件包的东西， `aptitude` 将保留它们。

附加题

好吧，东西真多。但这里还有更多： 键入 `aptiutde search emacs` 。弄清楚 `v` 的意思是什么。 阅读或浏览 Debian 手册中的第 2 章 Debian 软件包管理。

练习 15：系统启动：运行级别，`/etc/init.d`，`rcconf`，`update-rc.d`

> 原文：Exercise 15. System boot: runlevels, /etc/init.d, rcconf, update-rc.d
>
> 译者：飞龙
>
>
>
> 自豪地采用谷歌翻译

首先我会给出一个典型的系统启动过程的概述：

```
你
    按电源开关（或启动虚拟机）
    现在计算机获得控制权
    控制权传给了 BIOS
BIOS
    执行硬件特定的任务
    执行开机自检（POST），测试你的硬件
    检测安装的硬件，如硬盘，内存类型和数量，...
    通过将初始值写入其内存来初始化硬件
    找到一个启动设备，通常是一个硬盘
    读取并执行位于此磁盘开头的 MBR（主引导记录）
    控制权现在传给了 MBR
MBR
    MBR 寻找并执行 GRUB（多重操作系统启动管理器）
    控制权现在传给了 GRUB
GRUB
    查找可用的文件系统
    查找并读取其配置文件，来了解：
        系统位于哪里
        启动什么系统
        执行什么其他的操作
    执行 Linux 内核，Linux 操作系统的主要部分
    控制权现在传给了 Linux 内核
Linux 内核
    查找并加载 initrd，这是初始的 ram 磁盘
        initrd 包含必要驱动程序，允许真实文件系统的访问和挂载
    挂载文件系统，它在 GRUB 配置文件中指定。
    执行`/sbin/init`，一个启动所有其他程序的特殊程序
    控制权现在传给了 init
init
    查看`etc/inittab`来确定所需的运行级别
    加载适合此运行级别的所有程序
        加载来自`/etc/rc.d/rc2.d/`的所有程序，因为 2 是默认的 Debian
运行级别
        启动 SSH 和 TTY，以便你可以连接到你的计算机
    启动现在完成了
你
    使用 SSH 连接到你的计算机
    SSH 守护进程为你执行 bash shell
    你现在可以输入东西
    你再次获得控制权
```

现在我们只对“init”和“运行级别”阶段感兴趣，所以我将总结一下，系统如何启动并自动启动一些程序。首先，有一些术语：

- 守护进程 - 一直运行在后台的程序。这意味着它不在乎你是否登录系统，通常你不需要手动启动它，因为它在计算机启动时自动启动。
- 运行级别 - 系统运行模式。基本上，这只是一个数字，提供给 `init` 程序，它知道哪些守护程序与每个数字相关联，并根据需要启动并停止这些守护程序。

在 Debian 中有以下运行级别：

ID	描述
S	系统通电后会执行它
0	停止，这定义了当系统关闭时执行哪些操作。
1	单用户模式，这是一种特殊的故障排除模式。在这种模式下，大多数守护进程不会自动启动。
2~5	完全多用户，配置的守护程序在此模式下启动。
6	重启，类似停止，但不是关闭系统而是重新启动。

但是 `init` 怎么知道的？好吧，这是用于它的特殊目录。

```
user1@vm1:/etc$ find /etc -type d -name 'rc*' 2>/dev/null | sort
/etc/rc0.d
/etc/rc1.d
/etc/rc2.d
/etc/rc3.d
/etc/rc4.d
/etc/rc5.d
/etc/rc6.d
/etc/rcS.d
```

你可能能猜到，每个数字和 `S` 对应表中的运行级别。让我们列出其中一个目录，它在正常启动中启动所有所需的守护进程。

```
user1@vm1:/etc$ ls -al /etc/rc2.d | awk '{printf "%-15.15s %-3.3
s %s\n",$9,$10,$11}'
.
..
README
S14portmap      ->  ../init.d/portmap
S15nfs-common   ->  ../init.d/nfs-common
S17rsyslog      ->  ../init.d/rsyslog
S17sudo         ->  ../init.d/sudo
S18acpid        ->  ../init.d/acpid
S18atd          ->  ../init.d/atd
S18cron         ->  ../init.d/cron
S18exim4        ->  ../init.d/exim4
S18ssh          ->  ../init.d/ssh
S20bootlogs     ->  ../init.d/bootlogs
S21rc.local     ->  ../init.d/rc.local
S21rmnologin    ->  ../init.d/rmnologin
S21stop-bootlog ->  ../init.d/stop-bootlogd
```

如你所见，此目录中的文件只是实际启动脚本的符号链接。我们来看看其中一个链接： `S18ssh→../init.d/ssh` 。这是关于这个文件的事情：

- 它是一个 `./init.d/ssh` 文件的链接
- 它以 `S` 开始，意味着“启动”。Debian 启动系统中使用的每个脚本至少有 2 个参数，“启动”和“停止”。现在我们可以说，当我们的系统切换到运行级别 2 时，该脚本将使用动作“启动”来执行 。
- 它有一个数字 18。 `rc` 目录中的脚本以字典序执行，所以现在我们明白，在启动 `ssh` 之前 ，系统启动 `portmap` ， `nfs-common` ， `rsyslog` 和 `sudo` 。 `rsyslog` 是一个系统日志守护程序，特别是 `ssh` 想要记录谁在什么时候访问系统，所以在启动之前需要运行 `rsyslog` 。

现在，你将学习如何列出启用的服务（守护程序），以及启用和禁用服务（守护程序）。

这样做

```
sudo aptitude install rcconf
ls -al /etc/rc2.d
sudo rcconf --list
sudo update-rc.d exim4 disable
ls -al /etc/rc2.d
sudo rcconf --list
sudo update-rc.d exim4 enable
ls -al /etc/rc2.d
sudo rcconf --list
```

你会看到什么

```
user1@vm1:/var/log$ sudo aptitude install rcconf
The following NEW packages will be installed:
  rcconf
0 packages upgraded, 1 newly installed, 0 to remove and 0 not up
graded.
Need to get 0 B/23.9 kB of archives. After unpacking 135 kB will
 be used.
Selecting previously deselected package rcconf.
(Reading database ... 24239 files and directories currently inst
alled.)
Unpacking rcconf (from .../archives/rcconf_2.5_all.deb) ...
Processing triggers for man-db ...
Setting up rcconf (2.5) ...

user1@vm1:/etc$ ls -al /etc/rc2.d
total 12
drwxr-xr-x  2 root root 4096 Jun 27 11:42 .
drwxr-xr-x 68 root root 4096 Jun 25 18:43 ..
-rw-r--r--  1 root root  677 Mar 27 05:50 README
```

```
lrwxrwxrwx  1 root root   17 Jun  4 11:53 S14portmap -> ../init.
d/portmap
lrwxrwxrwx  1 root root   20 Jun  4 11:53 S15nfs-common -> ../in
it.d/nfs-common
lrwxrwxrwx  1 root root   17 Jun  4 11:53 S17rsyslog -> ../init.
d/rsyslog
lrwxrwxrwx  1 root root   14 Jun 15 19:02 S17sudo -> ../init.d/s
udo
lrwxrwxrwx  1 root root   15 Jun  4 11:53 S18acpid -> ../init.d/
acpid
lrwxrwxrwx  1 root root   13 Jun  4 11:53 S18atd -> ../init.d/at
d
lrwxrwxrwx  1 root root   14 Jun  4 11:53 S18cron -> ../init.d/c
ron
lrwxrwxrwx  1 root root   15 Jun 27 11:42 S18exim4 -> ../init.d/
exim4
lrwxrwxrwx  1 root root   13 Jun  4 11:53 S18ssh -> ../init.d/ss
h
lrwxrwxrwx  1 root root   18 Jun  4 11:53 S20bootlogs -> ../init
.d/bootlogs
lrwxrwxrwx  1 root root   18 Jun  4 11:53 S21rc.local -> ../init
.d/rc.local
lrwxrwxrwx  1 root root   19 Jun  4 11:53 S21rmnologin -> ../ini
t.d/rmnologin
lrwxrwxrwx  1 root root   23 Jun  4 11:53 S21stop-bootlogd -> ..
/init.d/stop-bootlogd
user1@vm1:/etc$ sudo rcconf --list
rsyslog on
ssh on
bootlogs on
portmap on
sudo on
nfs-common on
udev on
console-setup on
kbd on
exim4 on
keyboard-setup on
acpid on
cron on
atd on
procps on
module-init-tools on
user1@vm1:/etc$ sudo update-rc.d exim4 disable
update-rc.d: using dependency based boot sequencing
insserv: warning: current start runlevel(s) (empty) of script `e
xim4' overwrites defaults (2 3 4 5).
insserv: warning: current stop runlevel(s) (0 1 2 3 4 5 6) of sc
ript `exim4' overwrites defaults (0 1 6).
user1@vm1:/etc$ ls -al /etc/rc2.d
total 12
drwxr-xr-x  2 root root 4096 Jun 27 11:43 .
drwxr-xr-x 68 root root 4096 Jun 25 18:43 ..
```

```
lrwxrwxrwx  1 root root   15 Jun 27 11:43 K01exim4 -> ../init.d/
exim4
-rw-r--r--  1 root root  677 Mar 27 05:50 README
lrwxrwxrwx  1 root root   17 Jun  4 11:53 S14portmap -> ../init.
d/portmap
lrwxrwxrwx  1 root root   20 Jun  4 11:53 S15nfs-common -> ../in
it.d/nfs-common
lrwxrwxrwx  1 root root   17 Jun  4 11:53 S17rsyslog -> ../init.
d/rsyslog
lrwxrwxrwx  1 root root   14 Jun 15 19:02 S17sudo -> ../init.d/s
udo
lrwxrwxrwx  1 root root   15 Jun  4 11:53 S18acpid -> ../init.d/
acpid
lrwxrwxrwx  1 root root   13 Jun  4 11:53 S18atd -> ../init.d/at
d
lrwxrwxrwx  1 root root   14 Jun  4 11:53 S18cron -> ../init.d/c
ron
lrwxrwxrwx  1 root root   13 Jun  4 11:53 S18ssh -> ../init.d/ss
h
lrwxrwxrwx  1 root root   18 Jun  4 11:53 S20bootlogs -> ../init
.d/bootlogs
lrwxrwxrwx  1 root root   18 Jun  4 11:53 S21rc.local -> ../init
.d/rc.local
lrwxrwxrwx  1 root root   19 Jun  4 11:53 S21rmnologin -> ../ini
t.d/rmnologin
lrwxrwxrwx  1 root root   23 Jun  4 11:53 S21stop-bootlogd -> ..
/init.d/stop-bootlogd
user1@vm1:/etc$ sudo rcconf --list
rsyslog on
ssh on
bootlogs on
portmap on
sudo on
nfs-common on
udev on
console-setup on
kbd on
keyboard-setup on
acpid on
cron on
atd on
procps on
module-init-tools on
exim4 off
user1@vm1:/etc$ sudo update-rc.d exim4 enable
update-rc.d: using dependency based boot sequencing
user1@vm1:/etc$ ls -al /etc/rc2.d
total 12
drwxr-xr-x  2 root root 4096 Jun 27 11:43 .
drwxr-xr-x 68 root root 4096 Jun 25 18:43 ..
-rw-r--r--  1 root root  677 Mar 27 05:50 README
lrwxrwxrwx  1 root root   17 Jun  4 11:53 S14portmap -> ../init.
d/portmap
```

```
lrwxrwxrwx  1 root root   20 Jun  4 11:53 S15nfs-common -> ../in
it.d/nfs-common
lrwxrwxrwx  1 root root   17 Jun  4 11:53 S17rsyslog -> ../init.
d/rsyslog
lrwxrwxrwx  1 root root   14 Jun 15 19:02 S17sudo -> ../init.d/s
udo
lrwxrwxrwx  1 root root   15 Jun  4 11:53 S18acpid -> ../init.d/
acpid
lrwxrwxrwx  1 root root   13 Jun  4 11:53 S18atd -> ../init.d/at
d
lrwxrwxrwx  1 root root   14 Jun  4 11:53 S18cron -> ../init.d/c
ron
lrwxrwxrwx  1 root root   15 Jun 27 11:43 S18exim4 -> ../init.d/
exim4
lrwxrwxrwx  1 root root   13 Jun  4 11:53 S18ssh -> ../init.d/ss
h
lrwxrwxrwx  1 root root   18 Jun  4 11:53 S20bootlogs -> ../init
.d/bootlogs
lrwxrwxrwx  1 root root   18 Jun  4 11:53 S21rc.local -> ../init
.d/rc.local
lrwxrwxrwx  1 root root   19 Jun  4 11:53 S21rmnologin -> ../ini
t.d/rmnologin
lrwxrwxrwx  1 root root   23 Jun  4 11:53 S21stop-bootlogd -> ..
/init.d/stop-bootlogd
user1@vm1:/etc$ sudo rcconf --list
rsyslog on
ssh on
bootlogs on
portmap on
sudo on
nfs-common on
udev on
console-setup on
kbd on
exim4 on
keyboard-setup on
acpid on
cron on
atd on
procps on
module-init-tools on
user1@vm1:/etc$
```

解释

1. 安装 `rcconf` 包，让你轻松管理运行级别。
2. 打印包含运行级别 2 的启动脚本的目录。现在启用了邮件服务器 `exim4` 。
3. 仅仅打印出相同运行级别的服务。请注意，由于它们被视为系统服务，因此存在多个未显示的服务。 `rcconf –list –expert` 会把它们全部列出，以及更多的驻留在不同的运行级别上的服务。

4. 禁用邮件服务器 `exim4` 的自动启动。
5. 打印出包括运行级别 2 的启动脚本的目录。 `exim4` 启动脚本现在从 `S18exim4` 重命名为 `K01exim4` 。这意味着 `exim4` 进入此级别时已停止（被杀死）。如果 `exim4` 开始没有运行，就没有任何反应。
6. 打印运行级别 2 的服务。服务 `exim4` 现在已关闭。
7. 开启 `exim4` 的自动启动。
8. 再次打印包含运行级别 2 的启动脚本的目录， `exim4` 再次启动。
9. 打印运行级别 2 的服务。 `exim4` 的状态变更为已启动，和预期一样。

附加题

- 请阅读 Debian 启动过程：http://www.debian.org/doc/manuals/debian-reference/ch03.en.html
- 尝试这样做： `aptitude install sysv-rc-conf` ， `sysv-rc-conf -list` 。阅读 `man sysv-rc-conf` 。

练习 16：处理进程，`ps`，`kill`

> 原文：Exercise 16. Processes: working with proccesses, ps, kill
>
> 译者：飞龙
>
>
>
> 自豪地采用谷歌翻译

最简单的程序是硬盘上的文件，它包含中央处理器执行的指令。当你启动它的时候，它被复制到内存，控制权传递给它。被执行的程序称为进程。在例如 Linux 的多任务操作系统中，你可以启动程序的许多实例，因此可以从一个程序启动许多进程，所有程序将同时运行（执行）。

这是执行 `ls` 时发生的事情的概述：

```
你
   把 ls 和它的参数输入到你的终端模拟器，然后按 <ENTER>
   控制权现在传递给 Bash
Bash
   在你的硬盘上查找 ls
   将自身派生到 Bash 克隆体，也就是将自己克隆到内存中的新位置
   成为 Bash 克隆体的父进程
   控制权传给了传递给 Bash 克隆体
Bash 克隆体
   成为 Bash 的子进程
   保存 Bash 父进程的环境
   知道它是一个克隆体并且做出相应的反应
       使用 ls 覆盖自身
       控制权现在传递给 ls
ls
   为你打印一个目录列表或返回错误
   返回退出代码
   控制权现在传递给 Bash
Bash
   将 ls 退出代码赋给 ? 变量
   等待你的输入
你
   可以再次输入内容
```

一些进程不像 `ls` 那样交互，只是在后台静静地工作，就像 `ssh` 一样。进程有许多可能的状态，并且有许多操作，你可以通过信号机制对它们执行。

首先让我们谈论状态。如果你键入 `ps ax -forest`，它将打印出所有进程，你会得到这样的东西（跳过一些与硬件有关的进程）：

```
user1@vm1:/etc$ ps --forest ax
  PID TTY      STAT   TIME COMMAND
    1 ?        Ss     0:16 init [2]
  297 ?        S<s    0:00 udevd --daemon
  392 ?        S<     0:00  \_ udevd --daemon
  399 ?        S<     0:00  \_ udevd --daemon
  691 ?        Ss     0:00 /sbin/portmap
  703 ?        Ss     0:00 /sbin/rpc.statd
  862 ?        Sl     0:00 /usr/sbin/rsyslogd -c4
  886 ?        Ss     0:00 /usr/sbin/atd
  971 ?        Ss     0:00 /usr/sbin/acpid
  978 ?        Ss     0:01 /usr/sbin/cron
 1177 ?        Ss     0:00 /usr/sbin/sshd
 6671 ?        Ss     0:00  \_ sshd: user1 [priv]
 6675 ?        S      0:00      \_ sshd: user1@pts/0
 6676 pts/0    Ss     0:00          \_ -bash
 7932 pts/0    R+     0:00              \_ ps --forest ax
 1191 ?        Ss     0:00 /usr/sbin/exim4 -bd -q30m
 1210 tty2     Ss+    0:00 /sbin/getty 38400 tty2
 1211 tty3     Ss+    0:00 /sbin/getty 38400 tty3
 1212 tty4     Ss+    0:00 /sbin/getty 38400 tty4
 1213 tty5     Ss+    0:00 /sbin/getty 38400 tty5
 1214 tty6     Ss+    0:00 /sbin/getty 38400 tty6
 6216 tty1     Ss+    0:00 /sbin/getty 38400 tty1
```

让我们浏览这个列表：

`PID` - 进程 ID。每个进程都有与之相关联的唯一编号，用于唯一标识它。这意味着没有两个进程可以拥有相同的 PID。 `TTY` - 与进程相关联的电传模拟器，允许进程与你交换信息。 `STAT` - 当前进程状态。这将在下面详细描述。 `TIME` - 这是在 CPU 上执行此进程的时间（以分钟为单位）。 `COMMAND` - 这是带有参数的程序名称。请注意`/usr/sbin/sshd`是`sshd: user1`的父进程，而`sshd: user1`又是`sshd: user1@pts/0`的父进程，而`sshd: user1@pts/0`又是`bash`的父进程，而`bash`又是`ps –forest ax`的父进程。你需要更深入一些！

现在让我们讨论可能的进程状态。你可以参考`man ps`的 PROCESS STATE CODES。

状态	描述
D	不中断睡眠（通常为 IO）。进程繁忙或挂起，不响应信号，例如硬盘已经崩溃，读操作无法完成。
R	运行或可运行（在运行队列中）。进程正在执行中。
S	中断睡眠（等待事件完成）。例如，终端进程和 Bash 通常处于此状态，等待你键入某些内容。
T	停止，由任务控制信号或由于被追踪。
W	分页（从 2.6.xx 内核起无效，所以不用担心）。
X	死亡（不应该看到）。
Z	已停止（“僵尸”）进程，已终止，但未被父项收回。这种情况发生在错误终止的进程上。
<	高优先级（对其他用户不好）
N	低优先级（对其他用户很好）
L	将页面锁定到内存中（用于实时和自定义 IO）
s	是会话领导。Linux 中的相关进程被视为一个单元，并具有共享会话 ID（SID）。如果进程 ID（PID）= 会话 ID（SID），则此进程将是会话领导。
L	是多线程的（使用 CLONE_THREAD，例如 NPTL pthreads）
+	位于前台进程组。这样的处理器允许输入和输出到电传模拟器，tty。

现在让我们讨论，与所有这些进程沟通的方式。你可以通过发送信号来实现它。信号可能是一种鞭策进程方式，所以进程会听你的话，并做出你想要的行为。这是可能的进程的缩略列表，我从 `man kill` 的 `SIGNALS` 部分获得：

| 信号 | 编号 | 行为 | 描述 | | 0 | 0 | N/A | 退出代码表示是否可以发送信号 | | HUP | 1 | 退出 | 控制终端挂起或父进程死亡 | | INT | 2 | 退出 | 来自键盘的中断 | | QUIT | 3 | 内核 | 来自键盘的退出 | | ILL | 4 | 内核 | 非法指令 | | TRAP | 5 | 内核 | 跟踪/断点捕获 | | ABRT | 6 | 内核 | 来自 `abort(3)` 的中止信号 | | FPE | 8 | 内核 | 浮点异常 | | KILL | 9 | 退出 | 不可捕获，不可忽视的杀死 | | SEGV | 11 | 内核 | 内存引用无效 | | PIPE | 13 | 退出 | 损坏的管道：写入没有读者的管道 | | ALRM | 14 | 退出 | 来自 `alarm(2)` 的定时器信号 | | TERM | 15 | 退出 | 终止进程 |

同样，不要因为不理解而害怕，因为现在你只需要了解 `HUP`，`TERM` 和 `KILL`。甚至有一首关于 `KILL` 信号的歌曲，命名为[《Kill dash nine》]。另外，注意到 `abort(3)` 和 `alarm(2)` 了么？这意味着你可以通过键入 `man 3 abort` 和 `man 2 alarm` 来阅读相应的手册页。

现在，你将学习如何列出正在运行的进程并向其发送信号。

这样做

```
ps x
ps a
ps ax
ps axue --forest
dd if=/dev/zero of=~/test.img bs=1 count=$((1024*1024*1024))
&
kill -s USR1 $!
<ENTER>
kill -s USR1 $!
<ENTER>
kill -s TERM $!
<ENTER>
```

你会看到什么

```
user1@vm1:/etc$ ps x
  PID TTY      STAT   TIME COMMAND
 6675 ?        S      0:00 sshd: user1@pts/0
 6676 pts/0    Ss     0:00 -bash
 8193 pts/0    R+     0:00 ps x
user1@vm1:/etc$ ps a
  PID TTY      STAT   TIME COMMAND
 1210 tty2     Ss+    0:00 /sbin/getty 38400 tty2
 1211 tty3     Ss+    0:00 /sbin/getty 38400 tty3
 1212 tty4     Ss+    0:00 /sbin/getty 38400 tty4
 1213 tty5     Ss+    0:00 /sbin/getty 38400 tty5
 1214 tty6     Ss+    0:00 /sbin/getty 38400 tty6
 6216 tty1     Ss+    0:00 /sbin/getty 38400 tty1
 6676 pts/0    Ss     0:00 -bash
 8194 pts/0    R+     0:00 ps a
user1@vm1:/etc$ ps ax
  PID TTY      STAT   TIME COMMAND
    1 ?        Ss     0:16 init [2]
--- skipped --- skipped --- skipped ---
  691 ?        Ss     0:00 /sbin/portmap
  703 ?        Ss     0:00 /sbin/rpc.statd
  862 ?        Sl     0:00 /usr/sbin/rsyslogd -c4
  886 ?        Ss     0:00 /usr/sbin/atd
  971 ?        Ss     0:00 /usr/sbin/acpid
  978 ?        Ss     0:01 /usr/sbin/cron
 1177 ?        Ss     0:00 /usr/sbin/sshd
 1191 ?        Ss     0:00 /usr/sbin/exim4 -bd -q30m
 1210 tty2     Ss+    0:00 /sbin/getty 38400 tty2
 1211 tty3     Ss+    0:00 /sbin/getty 38400 tty3
 1212 tty4     Ss+    0:00 /sbin/getty 38400 tty4
 1213 tty5     Ss+    0:00 /sbin/getty 38400 tty5
```

```
 1214 tty6     Ss+    0:00 /sbin/getty 38400 tty6
 6216 tty1     Ss+    0:00 /sbin/getty 38400 tty1
 6671 ?        Ss     0:00 sshd: user1 [priv]
 6675 ?        S      0:00 sshd: user1@pts/0
 6676 pts/0    Ss     0:00 -bash
 8198 pts/0    R+     0:00 ps ax
user1@vm1:/etc$ ps axue --forest
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME
 COMMAND
--- skipped --- skipped --- skipped ---
root         1  0.0  0.0   8356   820 ?        Ss   Jun06   0:16
 init [2]
root       297  0.0  0.0  16976  1000 ?        S<s  Jun06   0:00
 udevd --daemon
root       392  0.0  0.0  16872   840 ?        S<   Jun06   0:00
  \_ udevd --daemon
root       399  0.0  0.0  16872   836 ?        S<   Jun06   0:00
  \_ udevd --daemon
daemon     691  0.0  0.0   8096   532 ?        Ss   Jun06   0:00
 /sbin/portmap
statd      703  0.0  0.0  14384   868 ?        Ss   Jun06   0:00
 /sbin/rpc.statd
root       862  0.0  0.1  54296  1664 ?        Sl   Jun06   0:00
 /usr/sbin/rsyslogd -c4
daemon     886  0.0  0.0  18716   436 ?        Ss   Jun06   0:00
 /usr/sbin/atd
root       971  0.0  0.0   3920   644 ?        Ss   Jun06   0:00
 /usr/sbin/acpid
root       978  0.0  0.0  22400   880 ?        Ss   Jun06   0:01
 /usr/sbin/cron
root      1177  0.0  0.1  49176  1136 ?        Ss   Jun06   0:00
 /usr/sbin/sshd
root      6671  0.0  0.3  70496  3284 ?        Ss   Jun26   0:00
  \_ sshd: user1 [priv]
user1     6675  0.0  0.1  70496  1584 ?        S    Jun26   0:00
      \_ sshd: user1@pts/0
user1     6676  0.0  0.6  23644  6536 pts/0    Ss   Jun26   0:00
          \_ -bash LANG=en_US.UTF-8 USER=user1 LOGNAME=user1 HOM
user1     8199  0.0  0.1  16312  1088 pts/0    R+   17:07   0:00
              \_ ps axue --forest TERM=screen-bce SHELL=/bin/bas
101       1191  0.0  0.1  44148  1076 ?        Ss   Jun06   0:00
 /usr/sbin/exim4 -bd -q30m
root      1210  0.0  0.0   5932   616 tty2     Ss+  Jun06   0:00
 /sbin/getty 38400 tty2
root      1211  0.0  0.0   5932   612 tty3     Ss+  Jun06   0:00
 /sbin/getty 38400 tty3
root      1212  0.0  0.0   5932   612 tty4     Ss+  Jun06   0:00
 /sbin/getty 38400 tty4
root      1213  0.0  0.0   5932   612 tty5     Ss+  Jun06   0:00
 /sbin/getty 38400 tty5
root      1214  0.0  0.0   5932   616 tty6     Ss+  Jun06   0:00
 /sbin/getty 38400 tty6
root      6216  0.0  0.0   5932   612 tty1     Ss+  Jun14   0:00
```

```
 /sbin/getty 38400 tty1
user1@vm1:/etc$ dd if=/dev/zero of=~/test.img bs=1 count=$((1024
*1024*1024)) &
[1] 8200
user1@vm1:/etc$ kill -s USR1 $!
user1@vm1:/etc$ 1455424+0 records in
1455424+0 records out
1455424 bytes (1.5 MB) copied, 1.76646 s, 824 kB/s

user1@vm1:/etc$ kill -s USR1 $!
user1@vm1:/etc$ 3263060+0 records in
3263060+0 records out
3263060 bytes (3.3 MB) copied, 3.94237 s, 828 kB/s

user1@vm1:/etc$ kill -s TERM $!
user1@vm1:/etc$
[1]+  Terminated              dd if=/dev/zero of=~/test.img bs=1
 count=$((1024*1024*1024))
user1@vm1:/etc$
```

解释

1. 打印你拥有（启动）的进程。
2. 仅打印与终端（ `tty` ）相关的进程和你拥有（启动）的进程。
3. 打印所有正在运行的进程。
4. 以树形式打印所有正在运行的进程，并包含附加信息，例如可用的相关用户名和环境。请注意，此信息仅适用于你拥有（启动）的进程。为了查看所有进程的环境信息，请输入 `sudo ps axue -forest` 。
5. 开始创建一个零填充文件（填充为空字符，现在不要纠结），并通过在末尾指定 `&` 发送到后台。
6. 查询 `dd` 的状态。
7. 因为 bash 只能打印一些东西来回应你的输入，你需要按 `<ENTER>` （发出空指令）。
8. 再次查询 `dd` 的状态。
9. 同样，你需要按 `<ENTER>` 来查看输出。
10. 向 `dd` 发送终止信号，所以 `dd` 退出了。
11. 为了看到它确实发生了，你需要再次按 `<ENTER>` 键 。

附加题

- 阅读 `man ps` ， `man kill` 。
- 阅读进程的生命周期，并研究这张图片：进程的工作流。
- 打印并填写信号表。你可以使用 kernel.org 中的文档。

练习 17：任务调度：`cron`，`at`

> 原文：Exercise 17. Job schedulers: cron, at
>
> 译者：飞龙
>
> 协议：CC BY-NC-SA 4.0
>
> 自豪地采用谷歌翻译

通常我们需要按计划执行程序。例如，让我们想象一下，你需要在每天的半夜备份你的作品。为了在 Linux 中完成它，有一个叫`cron`的特殊程序。这是一个恶魔，这意味着，当计算机启动后，它就是启动了，并在后台默默等待，在时机到来时为你执行其他程序。`cron`具有多个配置文件，系统级的，或者用户级的。默认情况下，用户没有`crontab`，因为没有为它们安排任何东西。这是`cron`配置文件的位置：

`/etc/crontab` - 系统级`cron`配置文件。`/var/spool/cron/crontabs/` - 用于存储用户配置文件的目录。

现在我们来谈谈`cron`配置文件的格式。如果你运行`cat /etc/crontab`，你将看到：

```
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username field
s,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bi
n

# m h dom mon dow user  command
17 *    * * *   root    cd / && run-parts --report /etc/cron.hou
rly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && r
un-parts --report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && r
un-parts --report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && r
un-parts --report /etc/cron.monthly )
#
```

它的语法足够简单，让我们选取一行：

```
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && r
un-parts --report /etc/cron.weekly
```

然后拆开：

```
17                                   # 每个小时的第 17 个分钟
*                                    # 月中的每一天
*                                    # 年中的每一月
*                                    # 每个星期
root                                 # 作为 root 用户
cd /                                 # 执行命令 'cd /'
&&                                   # 如果 'cd /' 执行成功，那么
run-parts --report /etc/cron.hourly # 执行命令 'run-parts --repor
t /etc/cron.hourly'
```

现在我总结 `cron` 的格式：

```
*    *    *    *    *     <用户>            <要执行的命令>
T    T    T    T    T     (仅仅用于系统
|    |    |    |    |      的 crontab)
|    |    |    |    |
|    |    |    |    +----- 星期几 (0 - 6) (0 是星期天, 或者使用名称)
|    |    |    +---------- 月份 (1 - 12)
|    |    +--------------- 天 (1 - 31)
|    +-------------------- 小时 (0 - 23)
+------------------------- 分钟 (0 - 59)
```

这是时间格式中，可能的字符的缩略列表：

- 星号（ `*` ）- 字段中的所有值，例如 分钟的 `*` 表示每分钟。
- 斜线（ `/` ）- 定义范围的增量。例如， `30-40/3` 意味着在第 30 分钟运行程序，每 3 分钟一次，直到第 40 分钟。
- 百分比（ `%` ）- 在命令字段中，百分比后的所有数据将作为标准输入发送到命令。现在不要纠结这个。
- 逗号（ `,` ）- 指定列表，例如分钟字段的 `30,40` 表示第 30 和 40`分钟。
- 连字（ `-` ）- 范围。例如，分钟字段的 `30-40` 意味着每分钟在30到40分钟之间。
- `L` - 指定最后一个东西，例如它允许你指定一个月的最后一天。

现在我会给你一些例子：

```
# m       h    dom  mon  dow    user            command
# （每月每天每小时）每分钟
*         *    *    *    *      root            /bin/false
# （每月每天）每小时的第 30~40 分钟中的每分钟
30-40     *    *    *    *      root            /bin/false
# （每月每天）每小时的第 30~40 分钟中的每五分钟
30-40/5   *    *    *    *      root            /bin/false
# （每月每天每小时）每五分钟
*/5       *    *    *    *      user            command to be exec
uted
# 每月的最后一天的每小时每分钟
*         *    L    *    *      root            /bin/false
# 每月的星期天和星期三的每小时每分钟
*         *    *    *    0,3    root          /bin/false
```

好的，但是如何加载 `crontab`？这是 `cron` 命令的列表：

- `crontab -l` - 打印出当前的 `crontab`。
- `crontab -e` - 为当前用户编辑 `crontab`。
- `crontab -r` - 删除当前用户的 `crontab`。
- `crontab /path/to/file` - 为当前用户加载 `crontab`，覆盖过程中的现有项。
- `crontab > /path/to/file` - 将 `crontab` 保存到文件中。

这就是如何使用 `cron` 系统守护进程。但是还有一个可以调度程序执行的选项。它就是 `at` 工具。它们之间的区别是，`cron` 为重复运行任务而设计，而且是很多次，并且 `at` 为调度一次性的任务而设计。这是相关的命令：

- `at` - 在指定的时间执行命令。
- `atq` - 列出待处理的任务。
- `atrm` - 删除任务。
- `batch` - 执行命令，然后系统空转。

这个信息转储似乎不够，现在我会给你用于 `at` 的，时间规范表格，取自 http://content.hccfl.edu/pollock/unix/atdemo.htm。在下面的例子中，假定的当前日期和时间是 2001 年 9 月 18 日星期二上午 10:00。

示例	含义
at noon	2001 年 9 月 18 日星期二下午 12:00
at midnight	2001 年 9 月 18 日星期二上午 12:00
at teatime	2001 年 9 月 18 日星期二下午 4:00
at tomorrow	2001 年 9 月 19 日星期二上午 10:00
at noon tomorrow	2001 年 9 月 19 日星期二下午 12:00
at next week	2001 年 9 月 25 日星期二上午 10:00
at next monday	2001 年 9 月 24 日星期二上午 10:00
at fri	2001 年 9 月 21 日星期二上午 10:00
at OCT	2001 年 10 月 18 日星期二上午 10:00
at 9:00 AM	2001 年 9 月 18 日星期二上午 9:00
at 2:30 PM	2001 年 9 月 18 日星期二下午 2:30
at 1430	2001 年 9 月 18 日星期二下午 2:30
at 2:30 PM tomorrow	2001 年 9 月 19 日星期二下午 2:30
at 2:30 PM next month	2001 年 10 月 18 日星期二下午 2:00
at 2:30 PM Fri	2001 年 9 月 21 日星期二下午 2:30
at 2:30 PM 9/21	2001 年 9 月 21 日星期二下午 2:30
at 2:30 PM Sept 21	2001 年 9 月 21 日星期二下午 2:30
at 2:30 PM 9/21/2010	2001 年 9 月 21 日星期二下午 2:30
at 2:30 PM 9.21.10	2001 年 9 月 21 日星期二下午 2:30
at now + 30 minutes	2001 年 9 月 18 日星期二上午 10:30
at now + 1 hour	2001 年 9 月 18 日星期二上午 11:00
at now + 2 days	2001 年 9 月 20 日星期二上午 10:00
at 4 PM + 2 days	2001 年 9 月 20 日星期二下午 4:00
at now + 3 weeks	2001 年 10 月 9 日星期二上午 10:00
at now + 4 months	2002 年 1 月 18 日星期二上午 10:00
at now + 5 years	2007 年 9 月 18 日星期二上午 10:00

现在你将学习如何添加、查看和移除 `at` 和 `crontab` 任务。

这样做

```
1: echo 'echo Here I am, sitting in ur at, staring at ur date: $
(date) | write user1' | at now + 1 minutes
2: atq
```

等待你的消息出现，按下 `<ENTER>` 并输入更多东西：

```
3: echo '* * * * * echo Here I am, sitting in ur crontab, starin
g at ur date: $(date) | write user1' > ~/crontab.tmp
4: crontab -l
5: crontab ~/crontab.tmp
6: crontab -l
```

现在等待这个消息出现并移除它。

```
7: crontab -r
8: crontab -l
```

你会看到什么

```
user1@vm1:~$ echo 'echo Here I am, sitting in ur at, staring at
ur date: $(date) | write user1' | at now + 1 minutes
warning: commands will be executed using /bin/sh
job 13 at Thu Jun 28 14:43:00 2012
user1@vm1:~$ atq
14      Thu Jun 28 14:45:00 2012 a user1
user1@vm1:~$
Message from user1@vm1 on (none) at 14:43 ...
Here I am, sitting in ur at, staring at ur date: Thu Jun 28 14:4
3:00 MSK 2012
EOF

user1@vm1:~$ crontab -l
no crontab for user1
user1@vm1:~$ echo '* * * * * echo Here I am, sitting in ur cront
ab, staring at ur date: $(date) | write user1' > ~/crontab.tmp
user1@vm1:~$ crontab -l
* * * * * echo Here I am, sitting in ur crontab, staring at ur d
ate: $(date) | write user1
user1@vm1:~$
Message from user1@vm1 on (none) at 14:47 ...
Here I am, sitting in ur crontab, staring at ur date: Thu Jun 28
 14:47:01 MSK 2012
EOF

user1@vm1:~$ crontab -r
user1@vm1:~$ crontab -l
no crontab for user1
user1@vm1:~$
```

解释

1. 让 `at` 在下一分钟执行命令 `echo Here I am, sitting in ur at, staring at ur date: $(date)` 。
2. 打印 `at` 的任务队列。
3. 将 `echo '* * * * * echo Here I am, sitting in ur crontab, staring a` 写入你的主目录中的 `crontab.tmp` 。
4. 打印你当前的 `crontab` ，但目前没有东西，所以它只是把这个告诉你。
5. 将 `crontab.tmp` 的内容加载到你的个人 `crontab` 文件。
6. 打印你当前的 `crontab` 。现在有一些东西。
7. 删除你当前的 `crontab` 。
8. 告诉你，你再次没有了 `crontab` 。

附加题

- 阅读`man crontab`，`man at`，`man write`。
- 让你的系统每 5 分钟告诉你当前时间。
- 让你的系统在每小时的开始告诉你当前时间。

练习 18：日志：`/var/log`，`rsyslog`，`logger`

> 原文：Exercise 18. Logging, /var/log, rsyslog, logger
>
> 译者：飞龙
>
> 协议：CC BY-NC-SA 4.0
>
> 自豪地采用谷歌翻译

守护进程是在后台运行的程序。所以问题来了：他们怎么告诉你他们在做什么？他们如何告诉你有什么问题？这个问题是由日志文件解决的，其中守护进程写入其状态和操作。在 Debian 中，这个文件位于 `/var/log` 目录下。

但谁写入这些文件？最明显的答案是守护进程本身，这实际上往往是错误的。在某些情况下，守护程序确实会自己编写日志文件，但通常它们通过名为 `rsyslogd` 的守护程序（称为 日志记录守护程序）来实现。它将日志写入不同的文件，来简化搜索和分析。为了区分这个文件，它有一个概念叫做“设施”。这是标准设施的列表：

设施	设施说明	设施	设施说明
auth	授权相关消息	LOCAL0	本地使用 0
authPriv	敏感的安全信息	LOCAL1	本地使用1
cron	Cron信息	local2	本地使用 2
daemon	系统守护程序	local3	本地使用 3
ftp	FTP 守护消息	local4	本地使用 4
kern	内核消息	local5	本地使用 5
lpr	行式打印机子系统	local6	当地使用 6
mail	邮件子系统	local7	当地使用 7
news	新闻子系统		
security	auth 的过时名称		
syslog	由 syslogd 内部生成的消息		
user			
uucp	UUCP 子系统		

每个条目也标记有严重性状态，以便分析发生了什么：

代码名称	严重性	描述	一般说明
alert	警报	必须立即采取行动。	应立即纠正，因此通知可以解决问题的人员。一个例子是丢失备用 ISP 连接。
crit	严重	严重情况。	应立即纠正，但表示主系统出现故障，一个例子就是主 ISP 连接的丢失 。
debug	调试	调试级别消息。	信息对开发人员有用，用于调试应用程序，在操作期间无用。
emerg	紧急	系统不可用	通常影响多个应用程序/服务器/站点的“紧急”状态。在这个级别，通常会通知所有技术人员。
err	错误	错误情况。	非紧急故障，应转发给开发人员或管理员；每个项目必须在给定的时间内解决。
error	错误	err 的弃用名称	---
info	信息	信息消息	正常操作的信息 - 可以用于收集报告，测量吞吐量等 - 无需采取任何行动。
notice	注意	正常但重要的状况。	不正常但不是错误情况的事件，可能汇总为邮件发给开发者或者管理员，来定位潜在问题 - 不需要立即采取行动。
panic	紧急	emerg 的弃用名称	---
warning	警告	警告情况。	警告消息，而不是错误，但表示如果不采取行动，将发生错误，例如文件系统 85% 占满 - 每个条目必须在给定时间内解决。
warn	警告	warning 的弃用名称	---

因为如果日志文件留给自己，它们往往会变得非常大，并且消耗所有可用的磁盘空间，所以有一种称为轮替的机制。默认情况下，这种机制通常只保留最后 7 天的日志文件，包括今天。轮替由 `logrotate` 守护进程执行，来帮助你了解这个守护进程做了什么。我为你将其写出来：

```
Day 0
    log.0 is created
Day 1
    mv log.0 log.1
    log.0 is created
Day 2
    mv log.1 log.2
    mv log.0 log.1
    log.0 is created
Day 3
    mv log.2 log.3
    mv log.1 log.2
    mv log.0 log.1
    log.0 is created
Day 4
    mv log.3 log.4
    mv log.2 log.3
    mv log.1 log.2
    mv log.0 log.1
    log.0 is created
Day 5
    mv log.4 log.5
    mv log.3 log.4
    mv log.2 log.3
    mv log.1 log.2
    mv log.0 log.1
    log.0 is created
Day 6
    mv log.5 log.6
    mv log.4 log.5
    mv log.3 log.4
    mv log.2 log.3
    mv log.1 log.2
    mv log.0 log.1
    log.0 is created
Day 7
    rm log.6
    mv log.5 log.6
    mv log.4 log.5
    mv log.3 log.4
    mv log.2 log.3
    mv log.1 log.2
    mv log.0 log.1
    log.0 is created
```

让我重复一下：

- 日志是一个记录时间的过程，使用自动化的计算机程序，来提供审计跟踪，可用于了解系统活动和诊断问题。
- 日志守护程序是程序，其他程序可能要求它在日志文件中写入内容。
- 每个日志条目具有设施（日志类别）和严重性 （它是多么重要）属性。

- 轮替是一个过程，仅保留有限数量的日志文件，来避免填满磁盘。
- 在 Debian 中，日志文件通常位于 `/var/log` 目录中。

这是处理日志的有用命令（要记住打开相关的手册页，并找出有什么选项）：

- `logger Hello, I have a kitty!` - 编写一个自定义日志消息。
- `ls -altr /var/log` - 列出日志目录，以这样一种方式，最后修改的文件到最后。
- `grep user1 /var/log/auth.log` - 列出文件中包含 `user1` 的所有行。
- `grep -irl user1 /var/log` - 列出所有包含 `user1` 的文件 。
- `find /var/log -mmin -10` - 找到在过去 10 分钟内被修改的任何文件。
- `tail /var/log/auth.log` - 打印日志文件的最后 10 行。
- `tail -f /var/log/auth.log` - 实时跟踪日志文件。配置守护进程时非常有用。

现在你将学习如何查看日志，并将一些东西写入系统日志。

这样做

```
 1: sudo -s
 2: cd /var/log
 3: ls -altr | tail
 4: tail auth.log
 5: grep user1 auth.log | tail
 6: /etc/init.d/exim4 restart
 7: find /var/log -mmin -5
 8: tail /var/log/exim4/mainlog
 9: grep -irl rcconf .
10: tail ./dpkg.log
11: last
12: lastlog
13: logger local0.alert I am a kitty, sittin in ur system watchi
n u work ^^
14: ls -altr | tail
15: tail messages
```

你会看到什么

```
user1@vm1:~$ sudo -s
root@vm1:/home/user1# cd /var/log
root@vm1:/var/log# ls -altr | tail
-rw-r-----  1 root        adm   46955 Jun 29 12:28 messages
-rw-r-----  1 root        adm   19744 Jun 29 12:28 dmesg
-rw-r-----  1 root        adm     696 Jun 29 12:28 daemon.log
drwxr-xr-x  7 root        root   4096 Jun 29 12:28 .
-rw-r-----  1 root        adm   60738 Jun 29 12:28 syslog
```

```
-rw-r-----  1 root        adm   58158 Jun 29 12:28 kern.log
-rw-r-----  1 root        adm   12652 Jun 29 12:28 debug
-rw-rw-r--  1 root        utmp  75264 Jun 29 12:28 wtmp
-rw-rw-r--  1 root        utmp 292584 Jun 29 12:28 lastlog
-rw-r-----  1 root        adm   38790 Jun 29 12:40 auth.log
root@vm1:/var/log# tail auth.log
Jun 29 12:28:22 vm1 sshd[983]: Server listening on 0.0.0.0 port
22.
Jun 29 12:28:22 vm1 sshd[983]: Server listening on :: port 22.
Jun 29 12:28:44 vm1 sshd[1214]: Accepted password for user1 from
 194.85.195.183 port 53775 ssh2
Jun 29 12:28:44 vm1 sshd[1214]: pam_unix(sshd:session): session
opened for user user1 by (uid=0)
Jun 29 12:30:49 vm1 sudo:    user1 : TTY=pts/0 ; PWD=/home/user1
 ; USER=root ; COMMAND=/bin/bash
Jun 29 12:30:53 vm1 login[1260]: pam_securetty(login:auth): unex
pected response from failed conversation function
Jun 29 12:30:53 vm1 login[1260]: pam_securetty(login:auth): cann
ot determine username
Jun 29 12:35:08 vm1 sudo:    user1 : TTY=pts/0 ; PWD=/home/user1
 ; USER=root ; COMMAND=/bin/bash
Jun 29 12:35:14 vm1 sudo:    user1 : TTY=pts/0 ; PWD=/home/user1
 ; USER=root ; COMMAND=/bin/bash
Jun 29 12:40:32 vm1 sudo:    user1 : TTY=pts/0 ; PWD=/home/user1
 ; USER=root ; COMMAND=/bin/bash
root@vm1:/var/log# tail auth.log | grep user1
Jun 29 12:28:44 vm1 sshd[1214]: Accepted password for user1 from
 194.85.195.183 port 53775 ssh2
Jun 29 12:28:44 vm1 sshd[1214]: pam_unix(sshd:session): session
opened for user user1 by (uid=0)
Jun 29 12:30:49 vm1 sudo:    user1 : TTY=pts/0 ; PWD=/home/user1
 ; USER=root ; COMMAND=/bin/bash
Jun 29 12:35:08 vm1 sudo:    user1 : TTY=pts/0 ; PWD=/home/user1
 ; USER=root ; COMMAND=/bin/bash
Jun 29 12:35:14 vm1 sudo:    user1 : TTY=pts/0 ; PWD=/home/user1
 ; USER=root ; COMMAND=/bin/bash
Jun 29 12:40:32 vm1 sudo:    user1 : TTY=pts/0 ; PWD=/home/user1
 ; USER=root ; COMMAND=/bin/bash
root@vm1:/var/log# grep user1 auth.log | tail
Jun 29 12:26:33 vm1 sshd[1302]: Accepted password for user1 from
 194.85.195.183 port 53008 ssh2
Jun 29 12:26:33 vm1 sshd[1302]: pam_unix(sshd:session): session
opened for user user1 by (uid=0)
Jun 29 12:26:38 vm1 sudo:    user1 : TTY=pts/0 ; PWD=/home/user1
 ; USER=root ; COMMAND=/bin/bash
Jun 29 12:28:02 vm1 sshd[1302]: pam_unix(sshd:session): session
closed for user user1
Jun 29 12:28:44 vm1 sshd[1214]: Accepted password for user1 from
 194.85.195.183 port 53775 ssh2
Jun 29 12:28:44 vm1 sshd[1214]: pam_unix(sshd:session): session
opened for user user1 by (uid=0)
Jun 29 12:30:49 vm1 sudo:    user1 : TTY=pts/0 ; PWD=/home/user1
 ; USER=root ; COMMAND=/bin/bash
```

```
Jun 29 12:35:08 vm1 sudo:    user1 : TTY=pts/0 ; PWD=/home/user1
 ; USER=root ; COMMAND=/bin/bash
Jun 29 12:35:14 vm1 sudo:    user1 : TTY=pts/0 ; PWD=/home/user1
 ; USER=root ; COMMAND=/bin/bash
Jun 29 12:40:32 vm1 sudo:    user1 : TTY=pts/0 ; PWD=/home/user1
 ; USER=root ; COMMAND=/bin/bash
root@vm1:/home/user1# /etc/init.d/exim4 restart
Stopping MTA for restart: exim4_listener.
Restarting MTA: exim4.
root@vm1:/home/user1# find /var/log -mmin -5
/var/log/exim4/mainlog
/var/log/auth.log
root@vm1:/home/user1# tail /var/log/exim4/mainlog
2012-06-29 12:24:11 exim 4.72 daemon started: pid=1159, -q30m, l
istening for SMTP on [127.0.0.1]:25 [::1]:25
2012-06-29 12:24:11 Start queue run: pid=1165
2012-06-29 12:24:11 End queue run: pid=1165
2012-06-29 12:28:22 exim 4.72 daemon started: pid=1190, -q30m, l
istening for SMTP on [127.0.0.1]:25 [::1]:25
2012-06-29 12:28:22 Start queue run: pid=1196
2012-06-29 12:28:22 End queue run: pid=1196
2012-06-29 12:41:18 exim 4.72 daemon started: pid=1622, -q30m, l
istening for SMTP on [127.0.0.1]:25 [::1]:25
2012-06-29 12:41:18 Start queue run: pid=1624
2012-06-29 12:41:18 End queue run: pid=1624
2012-06-29 12:42:28 exim 4.72 daemon started: pid=1886, -q30m, l
istening for SMTP on [127.0.0.1]:25 [::1]:25
root@vm1:/home/user1# grep -irl rcconf .
./aptitude
./apt/history.log
./apt/term.log
./dpkg.log
./auth.log
root@vm1:/home/user1# tail ./dpkg.log
2012-06-26 19:27:40 status unpacked rcconf 2.5
2012-06-26 19:27:40 status unpacked rcconf 2.5
2012-06-26 19:27:40 trigproc man-db 2.5.7-8 2.5.7-8
2012-06-26 19:27:40 status half-configured man-db 2.5.7-8
2012-06-26 19:27:40 status installed man-db 2.5.7-8
2012-06-26 19:27:41 startup packages configure
2012-06-26 19:27:41 configure rcconf 2.5 2.5
2012-06-26 19:27:41 status unpacked rcconf 2.5
2012-06-26 19:27:41 status half-configured rcconf 2.5
2012-06-26 19:27:41 status installed rcconf 2.5
root@vm1:/var/log# last
user1    pts/0        sis.site  Fri Jun 29 12:26   still logged
in
user1    pts/0        sis.site  Fri Jun 29 12:14 - down   (00:09
)
user1    pts/0        sis.site  Thu Jun 28 19:40 - 11:25  (15:45
)
user1    pts/0        sis.site  Wed Jun 27 19:14 - 17:04  (21:50
)
```

```
user1    pts/0        sis.site  Tue Jun 26 13:54 - 18:18 (1+04:2
3)
user1    pts/0        sis.site  Thu Jun 21 15:23 - 13:11 (4+21:4
7)
user1    pts/0        sis.site  Fri Jun 15 19:34 - 12:01 (5+16:2
6)
user1    pts/0        sis.site  Fri Jun 15 19:11 - 19:34  (00:22
)
reboot   system boot  2.6.32-5-amd64   Fri Jun 29 12:24 - 12:26
 (00:02)
user1    pts/0        sis.site  Fri Jun 29 12:14 - down   (00:09
)
root@vm1:/var/log# lastlog
Username         Port     From             Latest
root                                       **Never logged in**
daemon                                     **Never logged in**
bin                                        **Never logged in**
sys                                        **Never logged in**
sync                                       **Never logged in**
games                                      **Never logged in**
man                                        **Never logged in**
lp                                         **Never logged in**
mail                                       **Never logged in**
news                                       **Never logged in**
uucp                                       **Never logged in**
proxy                                      **Never logged in**
www-data                                   **Never logged in**
backup                                     **Never logged in**
list                                       **Never logged in**
irc                                        **Never logged in**
gnats                                      **Never logged in**
nobody                                     **Never logged in**
libuuid                                    **Never logged in**
Debian-exim                                **Never logged in**
statd                                      **Never logged in**
sshd                                       **Never logged in**
user1            pts/0    sis.site  Fri Jun 29 12:28:45 +0400 20
12
root@vm1:/var/log# logger local0.alert I am a kitty, sittin in u
r system watchin u work ^^
root@vm1:/var/log# ls -altr | tail
-rw-r-----  1 root        adm     696 Jun 29 12:28 daemon.log
drwxr-xr-x  7 root        root   4096 Jun 29 12:28 .
-rw-r-----  1 root        adm   58158 Jun 29 12:28 kern.log
-rw-r-----  1 root        adm   12652 Jun 29 12:28 debug
-rw-rw-r--  1 root        utmp  75264 Jun 29 12:28 wtmp
-rw-rw-r--  1 root        utmp 292584 Jun 29 12:28 lastlog
-rw-r-----  1 root        adm   38971 Jun 29 13:17 auth.log
-rw-r-----  1 root        adm     229 Jun 29 13:19 user.log
-rw-r-----  1 root        adm   60932 Jun 29 13:19 syslog
-rw-r-----  1 root        adm   47047 Jun 29 13:19 messages
root@vm1:/var/log# tail messages
Jun 29 12:28:21 vm1 kernel: [    1.846975] processor LNXCPU:00:
```

```
registered as cooling_device0
Jun 29 12:28:21 vm1 kernel: [    1.868828] usbcore: registered n
ew interface driver hiddev
Jun 29 12:28:21 vm1 kernel: [    1.895676] input: QEMU 0.14.1 QE
MU USB Tablet as /devices/pci0000:00/0000:00:01.2/usb1/1-1/1-1:1
.0/input/input4
Jun 29 12:28:21 vm1 kernel: [    1.895743] generic-usb 0003:0627
:0001.0001: input,hidraw0: USB HID v0.01 Pointer [QEMU 0.14.1 QE
MU USB Tablet] on usb-0000:00:01.2-1/input0
Jun 29 12:28:21 vm1 kernel: [    1.895762] usbcore: registered n
ew interface driver usbhid
Jun 29 12:28:21 vm1 kernel: [    1.895765] usbhid: v2.6:USB HID
core driver
Jun 29 12:28:21 vm1 kernel: [    2.373061] EXT3 FS on vda1, inte
rnal journal
Jun 29 12:28:21 vm1 kernel: [    2.394992] loop: module loaded
Jun 29 12:28:21 vm1 kernel: [    2.413478] input: ImExPS/2 Gener
ic Explorer Mouse as /devices/platform/i8042/serio1/input/input5
Jun 29 13:19:11 vm1 user1: local0.alert I am a kitty, sittin in
ur system watchin u work ^^
root@vm1:/var/log#
```

解释

- 打开 root（超级用户）shell。这是因为作为 `user1` 工作时，出于安全的考虑，你不能读取所有日志文件。
- 将目录更改为 `/var/log` 。
- 按日期排序打印所有文件，最后修改的文件在底部。
- 从 `auth.log` 打印最后 10 行，包含登录系统的信息。
- 从 `auth.log` 打印包含 `user1` 的最后 10 行。
- 重启 `exim4` 邮件服务器。
- 打印最近 5 分钟内的文件更改。现在，你可以轻松找到 `exim4` 在哪个文件中记录其操作。
- 从 `exim4` 日志打印出最后 10 行 。
- 在当前目录中的所有文件 搜索 `rcconf` 。现在，你可以轻松找到 Debian 包系统记录其操作的位置。
- 从 `dpkg.log` 打印最后 10 行，含有软件包安装和删除信息。
- 打印用户最后登录的信息。
- 打印所有用户最近登录的信息。
- 将你的消息传递给 `rsyslogd` 守护程序。
- 按日期排序打印所有文件，最后修改的文件位于底部。现在你可能会看到这里就是你的消息。
- 从消息中打印出最后10行，你可以看到你的消息确实已记录。

附加题

阅读 `rsyslogd` 和 `logger` 的手册页。通过阅读相应的手册页，找出 `last` 和 `lastlog` 之间的区别。阅读 `logrotate` 手册页并记住它的存在。执行 `tail -f /var/log/auth.log`，并生成 `vm1` 的第二个连接（如果你在 Windows 上工作，则为 putty）。不错吧？

练习 19：文件系统：挂载，`mount`，`/etc/fstab`

> 原文：Exercise 19. Filesystems: mounting, mount, /etc/fstab
>
> 译者：飞龙
>
> 协议：CC BY-NC-SA 4.0
>
> 自豪地采用谷歌翻译

我希望你熟悉分区的概念。如果没有，我会简要介绍一下。首先引用自维基百科：

> 磁盘分区是一种行为，将硬盘驱动器分为多个逻辑存储单元，它们被称为分区，来将一个物理磁盘驱动器视为多个磁盘。

看一看：

```
user1@vm1:~$ sudo parted /dev/vda
GNU Parted 2.3
Using /dev/vda
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) unit GB
(parted) p
Model: Virtio Block Device (virtblk)
Disk /dev/vda: 17.2GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos

Number  Start   End     Size    Type      File system     Flags
 1      0.00GB  13.3GB  13.3GB  extended
 5      0.00GB  1.02GB  1.02GB  logical   ext3            boot
 6      1.03GB  2.05GB  1.02GB  logical   linux-swap(v1)
 7      2.05GB  3.07GB  1.02GB  logical   ext3
 8      3.07GB  5.12GB  2.05GB  logical   ext3
 9      5.12GB  9.22GB  4.09GB  logical   ext3
10      9.22GB  13.3GB  4.09GB  logical   ext3

(parted)
```

这是一个物理硬盘，分为 7 个不同的分区。这样做的原因很多，但最好被理解为“分治”原则的应用。以这种方式分割时，流氓程序不能通过占用所有磁盘空间，使整个服务器崩溃，该程序将限制在其分区中。我不会再谈论磁盘分区，但是我会继续关注文件系统，再次引用维基百科：

> 文件系统是一种组织数据的手段。通过提供存储，检索和更新数据的过程，以及管理包含它的设备上的可用空间，数据预期在程序终止后保留。文件系统以有效的方式组织数据，并根据设备的特定特性进行调整。在操作系统和文件系统之间，通常存在紧耦合。一些文件系统提供了机制来控制数据和元数据的访问。确保可靠性是文件系统的主要职责。一些文件系统允许多个程序几乎同时更新同一个文件。
>
> 类 Unix 操作系统创建一个虚拟文件系统，这使得所有设备上的所有文件似乎都存在于单个层次结构中。这意味着，在这些系统中，有一个根目录，系统上存在的每个文件位于它下方的某个地方。类 Unix 系统可以使用 RAM 磁盘或网络共享资源作为其根目录。

这意味着，所有文件系统都集成在一个大树中。对于熟悉 Microsoft Windows 的人来说，这意味着比起 `C:\` 和 `D:\` 等盘符，这种命名方案有一个单独的根，`/`，所有其他分区都连接到它上面。将文件系统连接到现有目录的过程称为挂载。连接文件系统的目录称为挂载点。同样，看一看：

```
user1@vm1:~$ mount
/dev/vda5 on / type ext3 (rw,errors=remount-ro)
tmpfs on /lib/init/rw type tmpfs (rw,nosuid,mode=0755)
proc on /proc type proc (rw,noexec,nosuid,nodev)
sysfs on /sys type sysfs (rw,noexec,nosuid,nodev)
udev on /dev type tmpfs (rw,mode=0755)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
devpts on /dev/pts type devpts (rw,noexec,nosuid,gid=5,mode=620)
/dev/vda10 on /home type ext3 (rw)
/dev/vda7 on /tmp type ext3 (rw)
/dev/vda9 on /usr type ext3 (rw)
/dev/vda8 on /var type ext3 (rw)
```

这是我之前展示给你的相同分区，你可以在这个列表中看到挂载点。不以 `/dev/vda` 开头的是虚拟文件系统，它允许访问不同的系统设施，但它们和此练习无关。现在我们来看看 `/etc/fstab` 文件：

```
user1@vm1:~$ cat /etc/fstab
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to na
me devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point>   <type>  <options>       <dump>
<pass>
proc            /proc           proc    defaults        0       0
# / was on /dev/vda5 during installation
UUID=128559db-a2e0-4983-91ad-d4f43f27da49 /         ext3    erro
rs=remount-ro 0       1
# /home was on /dev/vda10 during installation
UUID=32852d29-ddee-4a8d-9b1e-f46569a6b897 /home     ext3    defa
ults        0       2
# /tmp was on /dev/vda7 during installation
UUID=869db6b4-aea0-4a25-8bd2-f0b53dd7a88e /tmp      ext3    defa
ults        0       2
# /usr was on /dev/vda9 during installation
UUID=0221be16-496b-4277-b131-2371ce097b44 /usr      ext3    defa
ults        0       2
# /var was on /dev/vda8 during installation
UUID=2db00f94-3605-4229-8813-0ee23ad8634e /var      ext3    defa
ults        0       2
# swap was on /dev/vda6 during installation
UUID=3a936af2-2c04-466d-b98d-09eacc5d104c none      swap    sw
            0       0
/dev/scd0       /media/cdrom0   udf,iso9660 user,noauto     0
    0
```

看起来很恐怖，但让我们选取一行：

```
# <file system>                             <mount point>  <type
>  <options>         <dump>  <pass>
UUID=128559db-a2e0-4983-91ad-d4f43f27da49   /              ext3
   errors=remount-ro 0       1
```

按照字段将其拆开。

```
UUID=128559db-a2e0-4983-91ad-d4f43f27da49 # Filesystem to mount.
 This UUID is synonim for /dev/vda5
/                     # This is root filesystem, mount it to /
ext3                  # This is ext3 filesystem. There are many d
ifferent filesystems out there
errors=remount-ro     # If any errors encountered during mounting
 filesystem should be remounted read-only
0                     # This filesystem should not be backed up b
y dump utility
1                     # This filesystem should be checked first b
y fsck utility
```

和之前一样，这些信息可以通过 `man fstab` 提供给你。现在我将向你展示使用现有文件系统的几个命令：

- `mount` - 打印出所有已挂载的文件系统。
- `mount -a` - 挂载 `/etc/fstab` 中描述的所有文件系统。
- `mount /dev/sda<N> /<mount point>` - 挂载分区。
- `umount /dev/sda<N> /<mount point>` - 解除挂载分区。
- `mount -h` - 打印出使用 `mount` 的简短帮助。
- `fsck` - 检查分区是否有错误。
- `blkid` - 打印出唯一的分区标识符。

现在，你将学习如何列出已安装的分区，挂载和解除挂载它们。

这样做

```
1: cat /etc/fstab
2: mount
3: sudo blkid
4: sudo umount /tmp
5: mount
6: sudo fsck /tmp
7: sudo mount -a
8: mount
```

你会看到什么

```
user1@vm1:~$ cat /etc/fstab
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to na
me devices
# that works even if disks are added and removed. See fstab(5).
```

```
#
# <file system> <mount point>   <type>  <options>       <dump>
<pass>
proc            /proc           proc    defaults        0
0
# / was on /dev/sda1 during installation
UUID=05d469bb-dbfe-4d5a-9bb2-9c0fe9fa8577 /               ext3
  errors=remount-ro 0       1
# /home was on /dev/sda9 during installation
UUID=a1b936a0-df38-4bf5-b095-6220ffdfc63c /home           ext3
  defaults        0       2
# /tmp was on /dev/sda8 during installation
UUID=d0a86453-0dbb-4f33-a023-6c09fe9fa202 /tmp ext3 defaults 0 2
# /usr was on /dev/sda5 during installation
UUID=b9544cbb-cdb6-4f3b-89e7-a339f52bfac7 /usr            ext3
  defaults        0       2
# /var was on /dev/sda6 during installation
UUID=e15e713b-5850-4bc3-b99e-ab6f1d037caa /var            ext3
  defaults        0       2
# swap was on /dev/sda7 during installation
UUID=4d516f09-80ff-4956-8a75-e9757697f6b1 none            swap
  sw              0       0
/dev/scd0       /media/cdrom0   udf,iso9660 user,noauto     0
    0
user1@vm1:~$ mount
/dev/sda1 on / type ext3 (rw,errors=remount-ro)
tmpfs on /lib/init/rw type tmpfs (rw,nosuid,mode=0755)
proc on /proc type proc (rw,noexec,nosuid,nodev)
sysfs on /sys type sysfs (rw,noexec,nosuid,nodev)
udev on /dev type tmpfs (rw,mode=0755)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
devpts on /dev/pts type devpts (rw,noexec,nosuid,gid=5,mode=620)
/dev/sda9 on /home type ext3 (rw)
/dev/sda5 on /usr type ext3 (rw)
/dev/sda6 on /var type ext3 (rw)
/dev/sda8 on /tmp type ext3 (rw)
/dev/sda8 on /tmp type ext3 (rw)
user1@vm1:~$ sudo blkid
/dev/sda1: UUID="05d469bb-dbfe-4d5a-9bb2-9c0fe9fa8577" TYPE="ext
3"
/dev/sda5: UUID="b9544cbb-cdb6-4f3b-89e7-a339f52bfac7" TYPE="ext
3"
/dev/sda6: UUID="e15e713b-5850-4bc3-b99e-ab6f1d037caa" TYPE="ext
3"
/dev/sda7: UUID="4d516f09-80ff-4956-8a75-e9757697f6b1" TYPE="swa
p"
/dev/sda8: UUID="d0a86453-0dbb-4f33-a023-6c09fe9fa202" TYPE="ext
3"
/dev/sda9: UUID="a1b936a0-df38-4bf5-b095-6220ffdfc63c" TYPE="ext
3"
user1@vm1:~$ sudo umount /tmp
user1@vm1:~$ mount
/dev/sda1 on / type ext3 (rw,errors=remount-ro)
```

```
tmpfs on /lib/init/rw type tmpfs (rw,nosuid,mode=0755)
proc on /proc type proc (rw,noexec,nosuid,nodev)
sysfs on /sys type sysfs (rw,noexec,nosuid,nodev)
udev on /dev type tmpfs (rw,mode=0755)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
devpts on /dev/pts type devpts (rw,noexec,nosuid,gid=5,mode=620)
/dev/sda9 on /home type ext3 (rw)
/dev/sda5 on /usr type ext3 (rw)
/dev/sda6 on /var type ext3 (rw)
user1@vm1:~$ sudo fsck /tmp
fsck from util-linux-ng 2.17.2
e2fsck 1.41.12 (17-May-2010)
/dev/sda8: clean, 11/61752 files, 13973/246784 blocks
user1@vm1:~$ sudo mount -a
user1@vm1:~$ mount
/dev/sda1 on / type ext3 (rw,errors=remount-ro)
tmpfs on /lib/init/rw type tmpfs (rw,nosuid,mode=0755)
proc on /proc type proc (rw,noexec,nosuid,nodev)
sysfs on /sys type sysfs (rw,noexec,nosuid,nodev)
udev on /dev type tmpfs (rw,mode=0755)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
devpts on /dev/pts type devpts (rw,noexec,nosuid,gid=5,mode=620)
/dev/sda9 on /home type ext3 (rw)
/dev/sda5 on /usr type ext3 (rw)
/dev/sda6 on /var type ext3 (rw)
/dev/sda8 on /tmp type ext3 (rw)
user1@vm1:~$
```

解释

1. 打印你的 `/etc/fstab` 文件的内容，它包含分区信息以及挂载位置。
2. 打印当前已挂载的分区。
3. 打印系统中所有分区的 UUID。
4. 解除挂载 `/tmp` 分区，以便你可以检查它。
5. 再次打印出当前已挂载的分区。 `/tmp` 现在不存在于此列表中。
6. 检查 `/tmp` 分区是否有错误。 `fsck` 通过读取相应的 `/etc/fstab` 条目知道要检查哪个分区。
7. 挂载 `/etc/fstab` 中描述的所有分区。
8. 再次打印当前已挂载的分区。 `/tmp` 已经返回了此列表。

附加题

- 阅读 `man fstab` , `man mount` 。
- 阅读 http://tldp.org/LDP/sag/html/filesystems.html 。

练习 20：文件系统：修改和创建文件系统，`tune2fs`，`mkfs`

> 原文：Exercise 20. Filesystems: modifying and creating filesystems, tune2fs, mkfs
>
> 译者：飞龙
>
> 协议：CC BY-NC-SA 4.0
>
> 自豪地采用谷歌翻译

让我来介绍一下文件系统相关的术语：

- 文件系统 - 一种组织数据的方式，通过提供存储，检索和更新数据的过程，以及管理包含它的设备上的可用空间，数据预期在终止后保留。
- Inode - 索引节点是一种结构，存储文件系统对象（文件，目录等）的所有信息，除数据内容和文件名之外。
- 块 - 可以分配的最小块磁盘空间。它通常默认为 4096 字节，或 4 千字节。
- 日志 - 一种结构，允许文件系统跟踪什么时候写入了什么。这样可以快速了解在断电或类似问题时，未正确写入的内容。

接下来，让我给大家简要介绍文件系统的工作原理。为了按名称访问文件，Linux 内核将：

- 在包含该文件的目录中查找文件名。
- 获取文件 Inode 号。
- 通过 Inode 区域中的数字查找 Inode。
- 读取此 Inode 的数据块的位置。
- 使用这个位置在数据区域中从这个块读取文件。

现在，每个文件系统都有很多与之相关的选项。这些选项可以通过 `tune2fs` 程序查看和更改。这是一个带注解的 `tune2fs -l /dev/sda8` 的输出：

```
user1@vm1:~$ sudo tune2fs -l /dev/sda8
tune2fs 1.41.12 (17-May-2010)
# 只是个标签，可以是任何东西，或者没有东西
Filesystem volume name:   <none>
# 这里应该是最后的挂载点
Last mounted on:          <not available>
# 唯一的文件系统标识符
Filesystem UUID:          869db6b4-aea0-4a25-8bd2-f0b53dd7a88e
# 已知位置的特殊数字，它定义了 FS 的类型，尝试这个：
# sudo dd if=/dev/sda8 of=/dev/stdout bs=1 skip=1080 count=2 | h
exdump -C
Filesystem magic number:  0xEF53
# FS 版本
Filesystem revision #:    1 (dynamic)
```

```
# 启用的 FS 功能
Filesystem features:
    # 这是一个日志文件系统。日志文件系统是一个文件系统，
    # 它跟踪日志中发生的变化（通常是一个循环日志，
    # 在文件系统的特定位置），在提交给主文件系统之前。
    # 在系统崩溃或者断电的事件中，这种文件系统能够更快
    # 恢复，并且不可能毁坏。
    # http://en.wikipedia.org/wiki/Journaling_file_system
    has_journal
    # 拥有扩展属性，例如扩展的 ACL。
    ext_attr
    # 为系统信息保留空间，这允许 FS 改变大小。
    resize_inode
    # 使用索引来加速大目录中的查找。
    dir_index
    # 在目录条目中储存文件类型信息。
    filetype
    # 意思是需要运行 fsck。
    needs_recovery
    # 更少的超级块备份，在大 FS 上节约空间。
    sparse_super
    # 是否可以包含 > 2GB 的文件。在创建 >2GB 的文件时，内核会自动设置它。
    large_file
# dir_index 中使用哪个哈希
Filesystem flags:          signed_directory_hash
# 挂载时使用什么选项
Default mount options:     (none)
# 是否需要执行 fsck
Filesystem state:          clean
# 错误时做什么：继续，以只读方式重新挂载，或者报错？
Errors behavior:           Continue
# 哪个 OS 使用这个 FS
Filesystem OS type:        Linux
# 索引节点总数。索引节点就是 "inode"。它的结构是：
# 储存所有文件系统对象（文件、目录，以及其他）的信息
# 除了文件内容和文件名称。也就是说，
# 你的文件数量不能多于索引节点数量。
# 这就是索引节点结构，它描述了里面储存了什么信息：
#/* 出现在 Inode 表中的 Inode 结构 */
#struct dinode
#{ ushort  di_mode;     /* mode and type of file   */
#  short   di_nlink;    /* number of links to file */
#  ushort  di_uid;      /* owner's user id         */
#  ushort  di_gid;      /* owner's group id        */
#  off_t   di_size;     /* number of bytes in file */
#  char    di_addr[39]; /* disk block addresses    */
#  char    di_gen;      /* file generation number  */
#  time_t  di_atime;    /* time last accessed      */
#  time_t  di_mtime;    /* time last modified      */
#  time_t  di_ctime;    /* time created            */
#};
# 这里也有很好的解释：
# http://support.tux4u.nl/dokuwiki/lib/exe/fetch.php?media=linux
```

```
:disk_bestandssysteem:inode.pdf
Inode count:              62464
# 当前有多少空闲节点
Free inodes:              62452
# 设备上的第一个块。由于每个分区都表示为单独的设备，
# 它设为 0。
First block:              0
# 这是文件系统中，第一个索引节点的节点号。
First inode:              11
# 索引节点的大小，以字节为单位。在新的 Linux 发行版中，这有时会默认增加，
# 为了允许文件中扩展属性的存储，例如，微秒时间戳。
Inode size:               256
# 添加索引节点字段所需的空间
Required extra isize:     28
# 添加索引节点字段要求的空间。不重要，因为这个大小
# 任何时候都是所需空间
Desired extra isize:      28
# 一个隐形节点，它储存文件系统的日志。
Journal inode:            8
# 块的总数。块是磁盘空间可分配的最小单位。
# 你可以使用下面的公式，以 GB 计算分区大小：
# 块的数量 * 块的大小
# ------------------------
#         1024^3
Block count:              249856
# 有多少块为超极用户保留。普通用户不能使用这个
# 保留空间。这是为了使系统保持运行，以防一些流氓软件
# 决定塞满所有可用的磁盘空间。
Reserved block count:     12492
# 当前有多少块是空闲的。
Free blocks:              241518
# 用于目录索引（dir_index）的算法。
Default directory hash:   half_md4
# 目前为止，我可以说，这是用于 dir_index 哈希算法的种子值。
Directory Hash Seed:      d02c6099-bd06-4d29-a3d7-779df2aa2410
# 日志备份选项。
Journal backup:           inode blocks
#
# 块的大小，以字节为单位。4096 字节就是 4 KB。
Block size:               4096
# 在 ex3 FS 中未实现。这是一个特性，它能够在一个块中写入多个小文件，
# 来节约空间。
Fragment size:            4096
# 保留的控件，所以组描述符表可能在未来会增长。
Reserved GDT blocks:      60
# 每个块组的块数量。块组包含文件系统重要的控制信息的冗余副本。
# （超级块和文件描述符），并包含一部分文件系统contains a
# （块的位图，索引节点的位图，一部分索引节点表，以及数据块）。
# 块组的结构在下表中展示：
#,---------+---------+---------+---------+---------+---------,
#| 超级     | FS       | 块的     | Inode    | Inode    | 数据     |
#| 块       | 描述符   | 位图     | 位图     | 表       | 块       |
#`---------+---------+---------+---------+---------+---------'
```

```
# http://tldp.org/HOWTO/Filesystems-HOWTO-6.html
Blocks per group:         32768
# 每个组的片段数量。因为 ext3 FS 中没有片段，
# 这等于每个组的块数量。
Fragments per group:      32768
# 每个组的索引节点数量。
Inodes per group:         7808
# 每个组的索引节点块。索引节点块是一个表的索引，
# 描述了所有文件属性，除了文件名称。它拥有数据块的索引。
# 数据块包含文件眞实内容。
# http://www.porcupine.org/forensics/forensic-discovery/chapter3
.html
Inode blocks per group:   488
# FS 的创建时间。
Filesystem created:       Mon Jul  2 06:16:24 2012
# 最后的 FS 挂载时间。
Last mount time:          Mon Jul  2 06:57:21 2012
# 最后的 FS 写入时间。
Last write time:          Mon Jul  2 06:57:21 2012
# FS 的挂载次数。
Mount count:              6
# 自动检查前的次数。如果文件系统的挂载次数是这个
# 或者检查间隔到了，那么 FS 会自动检查。
Maximum mount count:      34
# 最后的 fsck 执行时间
Last checked:             Mon Jul  2 06:16:24 2012
# 下一个 FS 检查间隔. 如果这个间隔到了，
# 或者到达了最大挂载数，FS 会自动检查。
Check interval:           15552000 (6 months)
# 下一个 FS 检查间隔，以人类可读的格式。
Next check after:         Sat Dec 29 05:16:24 2012
# 能够使用保留空间的用户的用户 ID。
# 它默认是 root 用户（超级用户）
Reserved blocks uid:      0 (user root)
# 能够使用保留空间的用户的组 ID。
# 它默认是 root 组
Reserved blocks gid:      0 (group root)
```

很可怕，是嘛？实际上你会发现，这个描述中只有几个参数实际上是有用的，它们是：

- 保留块数量。
- 最大挂载数。
- 检查间隔。

通常你不需要修改其他参数，默认情况下它们是正常的。以下是使用文件系统的命令列表：

- `mkfs.ext3` - 创建一个 `ext3` 文件系统。如果在具有现有文件系统的设备上执行此命令，则该文件系统将被销毁，因此请小心。
- `mkfs.ext4` - 创建一个 `ext4` 文件系统。这其实是相同的程序，尝

试 `sudo find /sbin -samefile sbin/mkfs.ext3` 。
- `tune2fs` - 打印并更改文件系统参数。

现在，你将学习如何创建新的文件系统并修改其参数。

这样做

```
1: sudo -s
2: umount /tmp
3: blkid | grep /dev/sda8
4: mkfs.ext3 /dev/sda8
5: blkid | grep /dev/sda8
6: blkid | grep /dev/sda8 >> /etc/fstab
7: vim /etc/fstab
```

现在你必须将 `/tmp` 那一行的 UUID。

```
# /tmp was on /dev/sda8 during installation
UUID=869db6b4-aea0-4a25-8bd2-f0b53dd7a88e /tmp            ext3
  defaults        0       2
```

替换为你添加到文件末尾的那个：

```
/dev/sda8: UUID="53eed507-18e8-4f71-9003-bcea8c4fd2dd" TYPE="ext
3" SEC_TYPE="ext2"
```

因为根据定义，你的 UUID 必须跟我的不同。在替换 UUID，编写文件，退出之后，继续并输入：

```
 8: mount /tmp
 9: tune2fs -c 2 /dev/sda8
10: unmount /tmp
11: fsck /tmp
12: for ((i=1;i<=4;i++)); do mount /tmp ; umount /tmp ; cat /var
/log/messages | tail -n 4 ; done
13: fsck /tmp
14: mount -a
```

你会看到什么

```
user1@vm1:~$ sudo -s
root@vm1:/home/user1# umount /tmp
```

```
root@vm1:/home/user1# blkid | grep /dev/sda8
/dev/sda8: UUID="869db6b4-aea0-4a25-8bd2-f0b53dd7a88e" TYPE="ext
3" SEC_TYPE="ext2"
root@vm1:/home/user1# mkfs.ext3 /dev/sda8
mke2fs 1.41.12 (17-May-2010)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
62464 inodes, 249856 blocks
12492 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=255852544
8 block groups
32768 blocks per group, 32768 fragments per group
7808 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376

Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 28 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to overri
de.
root@vm1:/home/user1# blkid | grep /dev/sda8
/dev/sda8: UUID="53eed507-18e8-4f71-9003-bcea8c4fd2dd" TYPE="ext
3" SEC_TYPE="ext2"
root@vm1:/home/user1# blkid | grep /dev/sda8 >> /etc/fstab
root@vm1:/home/user1# vim /etc/fstab
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point>   <type>  <options>       <dump>
<pass>
proc            /proc           proc    defaults        0
0
# / was on /dev/vda5 during installation
UUID=128559db-a2e0-4983-91ad-d4f43f27da49 /               ext3
  errors=re
# /home was on /dev/vda10 during installation
UUID=32852d29-ddee-4a8d-9b1e-f46569a6b897 /home           ext3
  defaults
# /tmp was on /dev/sda8 during installation
UUID=869db6b4-aea0-4a25-8bd2-f0b53dd7a88e /tmp            ext3
  defaults
# /usr was on /dev/vda9 during installation
UUID=0221be16-496b-4277-b131-2371ce097b44 /usr            ext3
  defaults
# /var was on /dev/vda8 during installation
UUID=2db00f94-3605-4229-8813-0ee23ad8634e /var            ext3
  defaults
```

```
# swap was on /dev/vda6 during installation
UUID=3a936af2-2c04-466d-b98d-09eacc5d104c none            swap
  sw
/dev/scd0       /media/cdrom0   udf,iso9660 user,noauto     0
    0
/dev/sda8: UUID="53eed507-18e8-4f71-9003-bcea8c4fd2dd" TYPE="ext
3" SEC_TYPE
                                                            22,1
       Bot

#
# <file system> <mount point>   <type>  <options>       <dump>
<pass>
proc            /proc           proc    defaults        0
0
# / was on /dev/vda5 during installation
UUID=128559db-a2e0-4983-91ad-d4f43f27da49 /               ext3
  errors=re
# /home was on /dev/vda10 during installation
UUID=32852d29-ddee-4a8d-9b1e-f46569a6b897 /home           ext3
  defaults
# /tmp was on /dev/sda8 during installation
UUID=53eed507-18e8-4f71-9003-bcea8c4fd2dd /tmp            ext3
  defaults
# /usr was on /dev/vda9 during installation
UUID=0221be16-496b-4277-b131-2371ce097b44 /usr            ext3
  defaults
# /var was on /dev/vda8 during installation
UUID=2db00f94-3605-4229-8813-0ee23ad8634e /var            ext3
  defaults

# swap was on /dev/vda6 during installation
UUID=3a936af2-2c04-466d-b98d-09eacc5d104c none            swap
  sw
/dev/scd0       /media/cdrom0   udf,iso9660 user,noauto     0
    0
"/etc/fstab" 22L, 1277C written
root@vm1:/home/user1# mount /tmp
root@vm1:/home/user1# tune2fs -c 2 /dev/sda8
tune2fs 1.41.12 (17-May-2010)
Setting maximal mount count to 2
root@vm1:/home/user1# unmount /tmp
root@vm1:/home/user1# fsck /tmp
fsck from util-linux-ng 2.17.2
e2fsck 1.41.12 (17-May-2010)
/dev/sda8: clean, 11/62464 files, 8337/249856 blocks (check in 2
 mounts)
root@vm1:/home/user1# for ((i=1;i<=4;i++)); do mount /tmp ; umou
nt /tmp ; cat /var/log/messages | tail -n 4 ; done
Jul  2 12:11:43 vm1 kernel: [21080.920658] EXT3-fs: mounted file
system with ordered data mode.
Jul  2 12:11:58 vm1 kernel: [21096.363787] kjournald starting.
Commit interval 5 seconds
```

```
Jul  2 12:11:58 vm1 kernel: [21096.364167] EXT3 FS on sda8, inte
rnal journal
Jul  2 12:11:58 vm1 kernel: [21096.364171] EXT3-fs: mounted file
system with ordered data mode.
Jul  2 12:11:58 vm1 kernel: [21096.364171] EXT3-fs: mounted file
system with ordered data mode.
Jul  2 12:11:58 vm1 kernel: [21096.381372] kjournald starting.
Commit interval 5 seconds
Jul  2 12:11:58 vm1 kernel: [21096.381539] EXT3 FS on sda8, inte
rnal journal
Jul  2 12:11:58 vm1 kernel: [21096.381542] EXT3-fs: mounted file
system with ordered data mode.
Jul  2 12:11:58 vm1 kernel: [21096.396152] kjournald starting.
Commit interval 5 seconds
Jul  2 12:11:58 vm1 kernel: [21096.396158] EXT3-fs warning: maxi
mal mount count reached, running e2fsck is recommended
Jul  2 12:11:58 vm1 kernel: [21096.396344] EXT3 FS on sda8, inte
rnal journal
Jul  2 12:11:58 vm1 kernel: [21096.396348] EXT3-fs: mounted file
system with ordered data mode.
Jul  2 12:11:58 vm1 kernel: [21096.412434] kjournald starting.
Commit interval 5 seconds
Jul  2 12:11:58 vm1 kernel: [21096.412441] EXT3-fs warning: maxi
mal mount count reached, running e2fsck is recommended
Jul  2 12:11:58 vm1 kernel: [21096.412610] EXT3 FS on sda8, inte
rnal journal
Jul  2 12:11:58 vm1 kernel: [21096.412612] EXT3-fs: mounted file
system with ordered data mode.
root@vm1:/home/user1# fsck /tmp
fsck from util-linux-ng 2.17.2
e2fsck 1.41.12 (17-May-2010)
/dev/sda8 has been mounted 4 times without being checked, check
forced.
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
/dev/sda8: 11/62464 files (0.0% non-contiguous), 8337/249856 blo
cks
root@vm1:/home/user1# mount -a
root@vm1:/home/user1#
```

解释

1. 执行 root（超级用户）shell。
2. 解除挂载 `/tmp` ，从 `/etc/fstab` 读取它的位置 。
3. 打印出 `/dev/sda8` 的UUID， `/dev/sda8` 是挂载在 `/tmp` 上的文件系统。
4. 在 `/dev/sda8` 上创建一个新的文件系统。
5. 再次打印出 `/dev/sda8` 的 UUID，注意如何变化，因为你创建了一个新的文

件系统。
6. 将此 UUID 附加到 `/etc/fstab` 。
7. 打开 `/etc/fstab` 进行编辑。
8. 挂载新创建的文件系统。这实际上是一个检查，是否你已经正确替换了 UUID，如果不是会有一个错误消息。
9. 设置每两次挂载检查 `/dev/sda8` 。
10. 解除挂载 `/dev/sda8` 。
11. 检查 `/dev/sda8` 。
12. 挂载，接触挂载 `/dev/sda8` ，并连续四次向你展示 `/var/log/messages/` 的最后4 行。请注意，从第三次开始，挂载系统通知你 需要运行 `e2fsck` 。如果你重新启动系统，它将为你运行 `e2fsck` 。
13. 检查 `/dev/sda8` 。 `fsck` 确定文件系统类型并自动调用 `e2fsck` 。
14. 挂载所有文件系统。如果没有错误，你已经完成了这个练习。

附加题

- 阅读 `man mkfs` ， `man mkfs.ext3` ， `man tune2fs` 。
- 阅读页面顶部的 `tune2fs -l` 列表，并为你的所有文件系统读取幻数。
- 手动计算文件系统的大小，使用在 `tune2fs -l` 列表的块描述中提供的公式。
- 阅读这个幻灯片，并完成它展示的东西：http://mcgrewsecurity.com/training/extx.pdf。

练习 21：文件系统：修改根目录，`chroot`

> 原文：Exercise 21. Filesystems: changing root directory, chroot
>
> 译者：飞龙
>
>
>
> 自豪地采用谷歌翻译

让我从另一个维基百科的引用开始：

> Unix 操作系统上的 `chroot` 是一个操作，可以为当前正在运行的进程及其进程修改根目录。在这种修改后的环境中运行的程序，不能指定（也就是访问）这个特定目录树之外的文件。术语 `chroot` 可以指 `chroot(2)` 系统调用或 `chroot(8)` 包装程序。修改后的环境称为 `chroot` 监牢。

这意味着你可以创建一个目录（例如 `/opt/root` ），将必要的程序复制到那里并执行此程序。对于这样的程序， `/opt/root/` 就是根目录 `/` 。要了解为什么你需要这样，请阅读维基百科 `chroot` 文章。

这是练习的时候了。你现在将使用 bash 创建一个最小的 `chroot` 环境。为此，你将创建一个目录结构，并将 bash 及其依赖项复制到其中。

现在，你将学习如何创建一个 `chroot` 环境并进入它。

这样做

```
sudo -s
ldd /bin/bash
mkdir -vp /opt/root/bin
mkdir -v /opt/root/lib
mkdir -v /opt/root/lib64
cp -v /bin/bash /opt/root/bin/
cp -v /lib/libncurses.so.5 /opt/root/lib/
cp -v /lib/libdl.so.2 /opt/root/lib
cp -v /lib/libc.so.6 /opt/root/lib
cp -v /lib64/ld-linux-x86-64.so.2 /opt/root/lib64
chroot /opt/root/
```

哇哦，你为你自己创建了一个 Linux，某种程度上是这样。

你会看到什么

```
user1@vm1:/opt~ sudo -s
root@vm1:/opt# ldd /bin/bash
        linux-vdso.so.1 =>  (0x00007fff17bff000)
        libncurses.so.5 => /lib/libncurses.so.5 (0x00007f4b1edc6
000)
        libdl.so.2 => /lib/libdl.so.2 (0x00007f4b1ebc2000)
        libc.so.6 => /lib/libc.so.6 (0x00007f4b1e85f000)
        /lib64/ld-linux-x86-64.so.2 (0x00007f4b1f012000)
root@vm1:/opt# mkdir -vp /opt/root/bin
mkdir: created directory `/opt/root'
mkdir: created directory `/opt/root/bin'
root@vm1:/opt# mkdir -v /opt/root/lib
mkdir: created directory `/opt/root/lib'
root@vm1:/opt# mkdir -v /opt/root/lib64
mkdir: created directory `/opt/root/lib64'
root@vm1:/opt# cp -v /bin/bash /opt/root/bin/
`/bin/bash' -> `/opt/root/bin/bash'
root@vm1:/opt# cp -v /lib/libncurses.so.5 /opt/root/lib/
`/lib/libncurses.so.5' -> `/opt/root/lib/libncurses.so.5'
root@vm1:/opt# cp -v /lib/libdl.so.2 /opt/root/lib
`/lib/libdl.so.2' -> `/opt/root/lib/libdl.so.2'
root@vm1:/opt# cp -v /lib/libc.so.6 /opt/root/lib
`/lib/libc.so.6' -> `/opt/root/lib/libc.so.6'
root@vm1:/opt# cp -v /lib64/ld-linux-x86-64.so.2 /opt/root/lib64
`/lib64/ld-linux-x86-64.so.2' -> `/opt/root/lib64/ld-linux-x86-6
4.so.2'
root@vm1:/opt# chroot /opt/root/
```

解释

1. 作为超级用户（root）执行 bash。
2. 打印出 bash 需要的的库。
3. 在一个命令中创建 `/opt/root/` 和 `/opt/root/bin/` 目录。很帅吧？
4. 创建 `/opt/root/lib` 目录。
5. 创建 `/opt/root/lib64` 目录。
6. 将 `/bin/bash` 复制到 `/opt/root/bin/` 。
7. 将 `/lib/libncurses.so.5` 复制到 `/opt/root/lib/` 。
8. 将 `/lib/libdl.so.2` 复制到 `/opt/root/lib/` 。
9. 将 `/lib/libc.so.6` 复制到 `/opt/root/lib/` 。
10. 将 `/lib64/ld-linux-x86-64.so.2` 复制到 `/opt/root/lib64/` 。
11. 将根目录更改为 `/opt/root/` 。

附加题

- 阅读 `man chroot` ， `man ldd` 。
- 将 `ls` 命令复制到你的 `chroot` 并使其正常工作。

- 一个难题：将 `vim` 复制到你的 `chroot` 并使其正常工作。

练习 22：文件系统：移动数据，`tar`，`dd`

> 原文：Exercise 22. Filesystems: moving data around: tar, dd
>
> 译者：飞龙
>
>
>
> 自豪地采用谷歌翻译

现在是时候自己看看了，Linux 中的所有东西只是一个文件。

这个练习是一个很大的练习，但是看看你学到了什么。完成之后，在 `man` 中查看所有故意不解释的程序参数，，并试图自己解释每个命令的作用。

现在你将学习如何玩转数据。

这样做

```
 1: tar -czvf root.tgz /opt/root/
 2: tar -tzvf root.tgz
 3: cd /tmp
 4: tar -zxvf ~/root.tgz
 5: ls -al
 6: dd_if=$(mount | grep /tmp | cut -d ' ' -f 1) && echo $dd_if
 7: sudo dd if=$dd_if of=~/tmp.img bs=10M
 8: cd && ls -alh
 9: sudo losetup /dev/loop1 ~/tmp.img && sudo mount /dev/loop1 /
mnt/
10: ls -al /mnt
11: sudo umount /mnt && sudo losetup -d /dev/loop1
12: sudo umount $dd_if && sudo mkfs.ext3 $dd_if
13: new_uuid=$(sudo tune2fs -l $dd_if | awk '/UUID/{print $3}')
&& echo $new_uuid
14: grep '/tmp' /etc/fstab
15: sed "s/^UUID=.*\/tmp\s\+ext3\s\+defaults\s\+[0-9]\s\+[0-9]\s
\?/UUID=$new_uuid \/tmp ext3 defaults 0 2/" /etc/fstab
```

现在使用 `sudo tune2fs -l` 和 `sudo blkid` 检查输出。如果 `/etc/fstab` 中的 UUID 替换看起来正常，执行实际的替换。

```
16: sudo sed -i'.bak' "s/^UUID=.*\/tmp\s\+ext3\s\+defaults\s\+[0
-9]\s\+[0-9]\s\?/UUID=$new_uuid \/tmp ext3 defaults 0 2/" /etc/f
stab
17: sudo mount -a && ls /tmp
18: sudo umount /tmp && pv ~/tmp.img | sudo dd of=$dd_if bs=10M
19: new_uuid=$(sudo tune2fs -l $dd_if | awk '/UUID/{print $3}')
&& echo $new_uuid
20: sudo sed -i'.bak' "s/^UUID=.*\/tmp\s\+ext3\s\+defaults\s\+[0
-9]\s\+[0-9]\s\?/UUID=$new_uuid \/tmp ext3 defaults 0 2/" /etc/f
stab
21: sudo mount -a
22: rm -v tmp.img
```

输入 y 并按下 <ENTER> 。

你会看到什么

```
user1@vm1:~$ tar -czvf root.tgz /opt/root/
tar: Removing leading '/' from member names
/opt/root/
/opt/root/bin/
/opt/root/bin/bash
/opt/root/lib64/
/opt/root/lib64/ld-linux-x86-64.so.2
/opt/root/lib/
/opt/root/lib/libdl.so.2
/opt/root/lib/libncurses.so.5
/opt/root/lib/libc.so.6
user1@vm1:~$ tar -tzvf root.tgz
drwxr-xr-x root/root         0 2012-07-05 03:14 opt/root/
drwxr-xr-x root/root         0 2012-07-05 03:14 opt/root/bin/
-rwxr-xr-x root/root    926536 2012-07-05 03:14 opt/root/bin/bas
h
drwxr-xr-x root/root         0 2012-07-05 03:14 opt/root/lib64/
-rwxr-xr-x root/root    128744 2012-07-05 03:14 opt/root/lib64/l
d-linux-x86-64.so.2
drwxr-xr-x root/root         0 2012-07-05 03:14 opt/root/lib/
-rw-r--r-- root/root     14696 2012-07-05 03:14 opt/root/lib/lib
dl.so.2
-rw-r--r-- root/root    286776 2012-07-05 03:14 opt/root/lib/lib
ncurses.so.5
-rwxr-xr-x root/root   1437064 2012-07-05 03:14 opt/root/lib/lib
c.so.6
user1@vm1:~$ cd /tmp
user1@vm1:/tmp$ tar -zxvf ~/root.tgz
opt/root/
opt/root/bin/
opt/root/bin/bash
opt/root/lib64/
```

```
opt/root/lib64/ld-linux-x86-64.so.2
opt/root/lib/
opt/root/lib/libdl.so.2
opt/root/lib/libncurses.so.5
opt/root/lib/libc.so.6
user1@vm1:/tmp$ ls -al
total 19
drwxrwxrwt  6 root  root   1024 Jul  5 04:17 .
drwxr-xr-x 22 root  root   1024 Jul  3 08:29 ..
drwxrwxrwt  2 root  root   1024 Jul  3 08:41 .ICE-unix
drwx------  2 root  root  12288 Jul  3 07:47 lost+found
drwxr-xr-x  3 user1 user1  1024 Jul  5 03:24 opt
-rw-r--r--  1 root  root    489 Jul  3 10:14 sources.list
-r--r-----  1 root  root    491 Jul  3 10:21 sudoers
drwxrwxrwt  2 root  root   1024 Jul  3 08:41 .X11-unix
user1@vm1:/tmp$ dd_if=$(mount | grep /tmp | cut -d ' ' -f 1) &&
echo $dd_if
/dev/sda8
user1@vm1:~$ cd && ls -alh
total 243M
drwxr-xr-x 3 user1 user1 4.0K Jul  5 04:27 .
drwxr-xr-x 4 root  root  4.0K Jul  3 08:39 ..
-rw------- 1 user1 user1   22 Jul  3 10:45 .bash_history
-rw-r--r-- 1 user1 user1  220 Jul  3 08:39 .bash_logout
-rw-r--r-- 1 user1 user1 3.2K Jul  3 08:39 .bashrc
-rw------- 1 user1 user1   52 Jul  5 04:12 .lesshst
drwxr-xr-x 3 user1 user1 4.0K Jul  5 03:23 opt
-rw-r--r-- 1 user1 user1  675 Jul  3 08:39 .profile
-rw-r--r-- 1 user1 user1 1.3M Jul  5 04:25 root.tgz
-rw-r--r-- 1 root  root  241M Jul  5 04:36 tmp.img
user1@vm1:~$ sudo losetup /dev/loop1 ~/tmp.img && sudo mount /de
v/loop1 /mnt/
user1@vm1:~$ ls -al /mnt
total 19
drwxrwxrwt  6 root  root   1024 Jul  5 04:17 .
drwxr-xr-x 22 root  root   1024 Jul  3 08:29 ..
drwxrwxrwt  2 root  root   1024 Jul  3 08:41 .ICE-unix
drwx------  2 root  root  12288 Jul  3 07:47 lost+found
drwxr-xr-x  3 user1 user1  1024 Jul  5 03:24 opt
-rw-r--r--  1 root  root    489 Jul  3 10:14 sources.list
-r--r-----  1 root  root    491 Jul  3 10:21 sudoers
drwxrwxrwt  2 root  root   1024 Jul  3 08:41 .X11-unix
user1@vm1:~$ sudo umount /mnt && sudo losetup -d /dev/loop1
user1@vm1:~$ sudo umount $dd_if && sudo mkfs.ext3 $dd_if
mke2fs 1.41.12 (17-May-2010)
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
Stride=0 blocks, Stripe width=0 blocks
61752 inodes, 246784 blocks
12339 blocks (5.00%) reserved for the super user
First data block=1
```

```
Maximum filesystem blocks=67371008
31 block groups
8192 blocks per group, 8192 fragments per group
1992 inodes per group
Superblock backups stored on blocks:
        8193, 24577, 40961, 57345, 73729, 204801, 221185

Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 27 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to overri
de.
user1@vm1:~$ new_uuid=$(sudo tune2fs -l $dd_if | awk '/UUID/{pri
nt $3}') && echo $new_uuid
f8288adc-3ef9-4a6e-aab2-92624276b8ba
user1@vm1:~$ grep '/tmp' /etc/fstab
# /tmp was on /dev/sda8 during installation
UUID=011b4530-e4a9-4d13-926b-48d9e33b64bf /tmp ext3 defaults 0 2
user1@vm1:~$ sed "s/^UUID=.*\/tmp\s\+ext3\s\+defaults\s\+[0-9]\s
\+[0-9]\s\?/UUID=$new_uuid \/tmp ext3 defaults 0 2/" /etc/fstab
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to na
me devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point>   <type>  <options>       <dump>
<pass>
proc            /proc           proc    defaults        0
0
# / was on /dev/sda1 during installation
UUID=91aacf33-0b35-474c-9c61-311e04b0bed1 /               ext3
  errors=remount-ro 0       1
# /home was on /dev/sda9 during installation
UUID=e27b0efb-8cf0-439c-9ebe-d59c927dd590 /home           ext3
  defaults        0       2
# /tmp was on /dev/sda8 during installation
UUID=f8288adc-3ef9-4a6e-aab2-92624276b8ba /tmp ext3 defaults 0 2
# /usr was on /dev/sda5 during installation
UUID=9f49821b-7f94-4915-b9a9-ed9f12bb6847 /usr            ext3
  defaults        0       2
# /var was on /dev/sda6 during installation
UUID=b7e908a1-a1cd-4d5c-bc79-c3a99d003e7c /var            ext3
  defaults        0       2
# swap was on /dev/sda7 during installation
UUID=292981d7-5a17-488f-8d9a-176b65f45d46 none            swap
  sw              0       0
/dev/scd0       /media/cdrom0   udf,iso9660 user,noauto     0
    0
sudo sed -i'.bak' "s/^UUID=.*\/tmp\s\+ext3\s\+defaults\s\+[0-9]\
```

```
s\+[0-9]\s\?/UUID=$new_uuid \/tmp ext3 defaults 0 2/" /etc/fstab
sudo mount -a && ls /tmp
user1@vm1:~$ sudo umount /tmp && pv ~/tmp.img | sudo dd of=$dd_i
f bs=10M
 241MB 0:00:04 [54.2MB/s] [======================================
================================================================
==========>] 100%
0+1928 records in
0+1928 records out
252706816 bytes (253 MB) copied, 5.52494 s, 45.7 MB/s
user1@vm1:~$ rm -v tmp.img
rm: remove write-protected regular file `tmp.img'? y
removed `tmp.img'
user1@vm1:~$
```

解释

1. 在你的主目录中创建归档或 `/opt/root/` 。归档文件的扩展名是 `.tgz` ，因为这个归档实际上由两部分组成，就像是俄罗斯套娃。第一部分由字母 `t` 指定，是一个大文件，其中所有归档文件由程序 `tar` 合并。第二部分由字母 `gz` 指定 ，意味着 `tar` 调用 `gzip` 程序来压缩它。
2. 测试这个归档。
3. 将目录更改为 `/tmp` 。
4. 解压你的归档。
5. 打印目录内容。
6. 提取挂载在 `/tmp` 上的分区的名称，将其存储在 `dd_if` 变量中，如果提取成功，打印出 `dd_if` 值。 `if` 代表输入文件。
7. 将整个分区复制到你的主目录中的 `tmp.img` 。dd 使用超级用户调用，因为你正在访问代表你的分区的文件 `/dev/sda8` ，该分区对普通用户不可访问。
8. 将目录更改为你的主目录并打印出其内容。
9. 告诉 Linux 将 `tmp.img` 文件用作（一种）物理分区并挂载它。
10. 打印出 `tmp.img` 的内容。你可以看到它真的是 `/tmp` 的精确副本 。
11.
12. 解除挂载 `tmp.img` ，并告诉 Linux 停止将其看做分区。
13. 解除挂载 `/tmp` 并在那里创建新的文件系统，删除该过程中的所有东西。
14. 提取你的新 `/tmp` 文件系统的UUID ，将其存储在 `new_uuid` 中，并打印出来。
15. 从 `/etc/fstab` 中打印描述旧的 `/tmp` 分区的一行。
16. 向你展示，修改后的 `/etc/fstab` 如何工作。通过使用正则表达式来完成，这个表达式用作掩码，定义了这一行：

    ```
    UUID=f8288adc-3ef9-4a6e-aab2-92624276b8ba /tmp ext3 defaults
     0 2
    ```

    完成这本书后，我会给你一个链接，让你学习如何创建这样的正则表达式。

17. 使用新的 UUID 实际替换 `/tmp` 旧的 UUID。
18. 挂载 `/etc/fstab` 中描述的所有文件系统，并列出新 `/tmp` 的内容
19. 解除挂载新的 `/tmp` 并从 `tmp.img` 恢复旧 `/tmp` 。
20. 获取旧 `/tmp` 的 UUID，它实际上与创建新文件系统之前相同，因为 `tmp.img` 是旧的 `/ tmp` 的完美副本。
21. 在 `/etc/fstab` 中用旧的 UUID 替换新的 UUID 。
22. 从 `/etc/fstab` 挂载所有文件系统。如果此命令不会导致错误，你可能一切正常。恭喜。
23. 从你的主目录中删除 `tmp.img` 。

附加题

- 尝试详细解释每个命令的作用。拿出一张纸，把它全部写出来。在 `man` 中查找在所有不能很好理解的命令和参数。
- 这个有些过早了，但为什么你能作为 `user1` 来发出删除命令，从你的主目录中删除 `tmp.img` ，考虑到 `tmp.img` 由 root 创建？

练习 23：文件系统：权限，`chown`，`chmod`，`umask`

> 原文：Exercise 23. Filesystems: security permissions, chown, chmod, umask
>
> 译者：飞龙
>
>
>
> 自豪地采用谷歌翻译

现在是时候了解 Linux 文件系统的安全模型了。我们首先引用维基百科的权限文章：

> 大多数当前文件系统拥有方法，来管理特定用户和用户组的权限或访问权的。这些系统控制用户查看或更改文件系统内容的能力。
>
> 类 Unix 系统的权限在三个不同的类中进行管理。这些类称为用户， 组和其他。实际上，Unix 权限是访问控制列表（ACL）的简化形式。
>
> 当在类 Unix 系统上创建新文件时，其权限将从创建它的进程的 umask 确定。

对于 Linux 中的每个文件，都有三个权限类。对于每个权限类，有三个权限。

这是权限类：

类	描述
用户	文件的拥有者。
分组	同组用户
其它人	任何其他用户或组

这是每个类可分配的权限：

权限	符号	描述
读	`r--`	读取文件的能力
写	`-w-`	写入文件的能力
执行	`--x`	将文件作为程序执行的能力，例如 ShellScript 应该设置这个

这两个表格应该总结一下：

所有者			同组			其它人		
`r`	`w`	`x`	`r`	`w`	`x`	`r`	`w`	`x`

这些权限表示为数字。考虑下面的输出：

```
user1@vm1:~$ ls -al tmp.img
-rw-r--r-- 1 root root 252706816 Jul  6 07:54 tmp.img
user1@vm1:~$ stat tmp.img
  File: 'tmp.img'
  Size: 252706816       Blocks: 494064     IO Block: 4096   regu
lar file
Device: 809h/2057d      Inode: 88534       Links: 1
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/
   root)
Access: 2012-07-06 07:56:58.000000000 -0400
Modify: 2012-07-06 07:54:54.000000000 -0400
Change: 2012-07-06 07:54:54.000000000 -0400
user1@vm1:~$
```

这里我们能够看到，`tmp.img` 由用户 `root`，分组 `root` 拥有，并且拥有权限 `-rw-r–r–` 。让我们试着阅读他们。

```
-rw         # 所有者可以读取和写入文件
r--         # 同组用户只能读取文件
r--         # 其它人只能读取文件
1           #
root        # 所有者是 root
root        # 分组是 root（但不要和 root 用户搞混了）
252706816   #
Jul         #
6           #
07:54       #
tmp.img     #
```

这里是八进制表示法的相同权限：

```
Access:
(
  0
  6 -rw
  4 r--
  4 ---
)
Uid: (    0/    root)
Gid: (    0/    root)
```

这是用于将八进制转换成符号的表格。

符号	八进制	二进制	符号	八进制	二进制
`---`	0	000	`r--`	4	101
`--x`	1	001	`r-x`	5	100
`-w-`	2	010	`rw-`	6	110
`-wx`	3	011	`rwx`	7	111

请注意，产生权限是通过简单相加获得的。例如，让我们获得 `rx` 权限。 在八进制符号中的 `r` 为 4， `x` 为 1， `1 + 4` 为 5，为 `rx` 。

现在让我们讨论状态输出 `0644` 中的零。这是为了设置一些叫做 SUID，SGID 和粘连位的东西。我不会详细介绍，但我会给你一个额外的附加题和翻译表。

特殊位：

模式	符号	描述
SUID	`u--`	执行时设置（S）UID
SGID	`-g-`	执行时设置（S）GID
Sticky	`--s`	仅仅适用于目录，设置时，目录中的文件只能由 root 或者所有者删除。

将这些特殊位转换为八进制记法：

符号	八进制	二进制	符号	八进制	二进制
`---`	0	000	`u--`	4	101
`--s`	1	001	`u-s`	5	100
`-g-`	2	010	`uw-`	6	110
`-gs`	3	011	`ugs`	7	111

那么新创建的文件呢？例如，你使用 `touch umask.test` 创建了一个文件，它将具有哪些权限？事实证明，你可以使用文件模式创建掩码（umask）来控制 。umask 是一种机制，在创建文件时定义哪些权限分配给文件。umask 通过 屏蔽来实现，即从默认值中减去权限，对于 bash 是 777，对于目录和文件是 666。Umask 也是为用户，组和其他人定义的。

映射 umask 值和权限：

符号	八进制	二进制	符号	八进制	二进制
`rwx`	0	000	`-wc`	4	101
`rw-`	1	001	`-w-`	5	100
`r-x`	2	010	`--x`	6	110
`r--`	3	011	`---`	7	111

为了更清楚地了解，这里是另一张表。请记住，这个权限被屏蔽掉，就是删除它们。为了简化本示例，用户，分组 和其他人的权限是一样的。

umask 值	屏蔽（移除）的权限	新文件的有效权限	注解
000	无	777 读写执行	保留所有默认权限
111	只执行	666 读和写	因为新文件不可执行
222	只写	555 读和执行	-
333	写和执行	444 只读	-
444	只读	333 写和执行	-
555	读和执行	222 只写	-
666	读和写	111 只执行	-
777	读写执行	000 无	不保留任何权限

另一个 umask 示例：

	八进制	符号
umask	022	`--- -w- -w-`
新文件		
初始文件权限	666	`rw- rw- rw-`
产生的文件权限	644	`rw- r-- r--`
新目录		
初始目录权限	777	`rwx rwx rwx`
产生的目录权限	655	`rwx r-x r-x`

让我们总结一下这个项目：

- 权限或访问权 - 控制文件和目录访问的机制。
- 权限模式 - 允许文件操作的权限类型。
    - 读取， `r` 读取文件的能力。

  - 写入，`w` - 写入文件的能力。
  - 执行，`x` - 作为程序执行文件的能力。对于目录，这具有特殊的含义，即它允许进入目录。
- 用户类 - 应用权限的实体。
  - 用户/所有者类，`u` - 文件或目录的所有者，通常是创建它们的人。
  - 分组类，`g` - 组是用户的集合。
  - 其他类，`o` - 除所有者和分组之外的所有人。
- Umask - 控制新创建文件的访问权的机制。

以及管理权限的命令：

- `chmod` — 修改文件权限
- `chown` — 修改所有者
- `umask` — 修改掩码，以便将权限赋予新创建的文件

现在你将学习如何修改文件权限，文件所有者和 umask。

这样做

```
umask
echo 'test' > perms.022
ls -l perms.022
stat perms.022  | grep 'Access: ('
chmod 000 perms.022
ls -al perms.0022
echo 'test' > perms.022
rm -v perms.022
```

记得上个练习的附加题中的问题吗？你现在处于类似的情况，因为你不能对此文件执行任何操作。但是为什么允许你删除它？这是因为当删除文件时，实际上是从目录中删除此文件的信息，对文件本身不做任何事情。我在这个话题上有很多的附加题。

```
umask 666
echo 'test' > perms.000
ls -l perms.000
cat perms.000
chmod 600 perms.000
cat perms.000
rm -v perms.000
umask 027
echo 'test' > perms.027
ls -l perms.027
sudo chown root perms.027
echo 'test1' >> perms.027
chown user1 perms.027
sudo chown user1 perms.027
echo 'test1' >> perms.027
rm -v perms.027
umask 022
```

你会看到什么

```
user1@vm1:~$ umask
0027
user1@vm1:~$ echo 'test' > perms.022
user1@vm1:~$ ls -l perms.022
-rw-r----- 1 user1 user1 5 Jul  9 10:23 perms.022
user1@vm1:~$ stat perms.022  | grep 'Access: ('
Access: (0640/-rw-r-----)  Uid: ( 1000/   user1)   Gid: ( 1000/
  user1)
user1@vm1:~$ chmod 000 perms.022
user1@vm1:~$ ls -al perms.0022
ls: cannot access perms.0022: No such file or directory
user1@vm1:~$ echo 'test' > perms.022
-bash: perms.022: Permission denied
user1@vm1:~$ rm -v perms.022
rm: remove write-protected regular file `perms.022'? y
removed `perms.022'
user1@vm1:~$ umask 666
user1@vm1:~$ echo 'test' > perms.000
user1@vm1:~$ ls -l perms.000
---------- 1 user1 user1 5 Jul  9 10:23 perms.000
user1@vm1:~$ cat perms.000
cat: perms.000: Permission denied
user1@vm1:~$ chmod 600 perms.000
user1@vm1:~$ cat perms.000
test
user1@vm1:~$ rm -v perms.000
removed `perms.000'
user1@vm1:~$ umask 027
user1@vm1:~$ echo 'test' > perms.027
user1@vm1:~$ ls -l perms.027
-rw-r----- 1 user1 user1 5 Jul  9 10:24 perms.027
user1@vm1:~$ sudo chown root perms.027
user1@vm1:~$ echo 'test1' >> perms.027
-bash: perms.027: Permission denied
user1@vm1:~$ chown user1 perms.027
chown: changing ownership of `perms.027': Operation not permitte
d
user1@vm1:~$ sudo chown user1 perms.027
user1@vm1:~$ echo 'test1' >> perms.027
user1@vm1:~$ rm -v perms.027
removed `perms.027'
user1@vm1:~$ umask 022
```

解释

1. 打印当前的 umask。
2. 创建 `perms.022` ，包含一行 `test` 。
3. 打印此文件的信息。
4. 以八进制表示法打印该文件的权限信息。

5. 更改此文件的权限，禁止任何人对此进行任何操作。
6. 打印此文件的信息。
7. 尝试用 `test` 替换此文件内容，由于缺少权限而失败。
8. 删除此文件。这是可能的，因为没有触碰文件本身，只有目录 `/home/user1` 中的条目。
9. 更改 umask，默认情况下不分配任何权限。
10. 创建 `perms.000` ，包含一行 `test` 。
11. 打印此文件的信息。
12. 试图打印出这个文件内容，这显然会导致错误。
13. 更改文件权限，来允许所有者读写。
14. 打印此文件内容，这次成功了。
15. 删除此文件。
16. 再次更改 umask
17. 创建 `perms.027` ，包含一行 `test` 。
18. 打印此文件的信息。
19. 将文件所有者更改为 root。
20. 尝试向文件追加一行 `test1` ，导致错误。
21. 尝试将文件所有者更改回 `user1` ，因为文件所有者的信息包含在文件本身而失败，更准确地说在其索引节点中。
22. 将文件所有者更改回 `user1` ，这次成功运行，因为以 root 身份运行。
23. 将一行 `test1` 添加到我们的文件，这次成功了。
24. 删除 `perms.027` 。
25. 将 umask 还原到其默认值。

附加题

- 读 `man chmod` ， `man chown` ， `man umask` 。
- 重新阅读 `man chmod` 中的 `setuid` ， `setgid` 和 `sticky` 位。这样设置你的目录的 `setuid` 位，执行 `umask 002 && echo test | sudo tee perms.root user1` 的时候，它是 `perms.root` 分组的结果。
- 弄清楚为什么 `umask 002` 不工作。
- 尝试这个：

  ```
  user1_block0=$(echo 'stat /user1' | sudo debugfs /dev/sda9 2
  >/dev/null | grep '(0)' | cut -d':' -f2)
  echo $user1_block0
  sudo dd if=/dev/sda9 bs=4096 skip=$user1_block0 count=1 | he
  xdump -C
  ```

  很酷吧？你刚刚从 `raw` 分区直接读取目录内容。那么当你删除文件时，就是从这里删除一个条目，你有权修改这个条目，因为这就是实际的目录（一个特殊的文件）。

练习 24：接口配置，`ifconfig`，`netstat`，`iproute2`，`ss`，`route`

> 原文：Exercise 24. Networking: interface configuration, ifconfig, netstat, iproute2, ss, route
>
> 译者：飞龙
>
>
>
> 自豪地采用谷歌翻译

这个练习对于你来说是一个很大的信息量，如果你不熟悉网络，这就是一个伤害。如果你感到非常失落，请立即跳到“这样做”的部分，并完成它。为了正确理解这部分，你至少应该非常熟悉网络的以下基本概念：

- 通信协议 - 通信协议，就是数字消息格式和规则的系统，用于在计算系统或在电子通讯中交换那些消息。
- 以太网 - 用于局域网（LAN）的计算机网络技术族。
- MAC 地址 - 分配给物理网段上通信的网络接口的唯一标识符。例如：`08:00:27:d4:45:68`。
- TCP/IP - 互联网协议套件是一组通信协议，用于互联网和类似网络，通常是广域网最流行的协议栈。它通常被称为 TCP/IP，由于其最重要的协议：传输控制协议（TCP）和互联网协议（IP）
- IP - 互联网协议（IP）是主要通信协议，用于跨互联网络中继转发数据报（也称为网络封包）。
- IP 地址 - 互联网协议地址。示例：`10.0.2.15`
- 端口) - 应用特定或流程特定的软件结构，在计算机的主机操作系统中用作通信端点。示例：`22`
- 网络套接字 - 跨计算机网络的，进程间通信流的端点。今天，大多数计算机之间的通信基于互联网协议；因此大多数网络套接字都是互联网套接字。
- 本地套接字地址 - 本地 IP 地址和端口号，例如：`10.0.2.15:22`。
- 远程套接字地址 - 远程 IP 地址和端口号，仅适用于已建立的 TCP 套接字。示例：`10.0.2.2:52173`。
- 套接字对 - 沟通本地和远程套接字，只有 TCP 协议。示例：`(10.0.2.15:22, 10.0.2.2:52173)`。
- 子网掩码 - 逻辑可见的 IP 网络细分。示例：`/24` 或另一个记号 `255.255.255.0`。
- 路由 - 在网络中选择路径，来发送网络流量的过程。
- 默认网关 - 在计算机网络中，网关是一个 TCP/IP 网络上的节点（路由器），作为另一个网络的接入点。默认网关是计算机网络上的节点，当 IP 地址与路由表中的任何其他路由不匹配时，网络软件使用它。示例：`10.0.2.2`。
- 广播地址 - 逻辑地址，其中连接到多重访问的网络的设备能接收数据报。发给广播地址的消息，通常会由所有附加到网络的主机接收，而不是特定主机。示

  例：`10.0.2.255`。
- ICMP - 互联网消息控制协议，示例用法：`ping 10.0.2.2`。
- TCP - 传输控制协议。在数据交换之前建立连接，因此设计上可靠。示例：SSH, HTTP。
- UDP - 用户数据报协议。传输数据而不建立连接，因此设计上不可靠。示例：DNS。

如果你对某些概念不熟悉，不用担心。

- 阅读相应的维基百科文章，直到你至少充分理解（但是深入钻研当然更好）。
- 观看 http://www.visualland.net.cn/ 的视频：
  - 展开站点左侧的 IP 地址树节点，并通过它来以你的方式实现。
  - 展开 TCP 树节点并执行相同操作。
- 阅读 Linux 网络概念介绍。这本指南很好，因为它甚至承认 互联网是为情欲而生的。

让我们继续。这是 Linux 网络相关的命令列表：

- `ifconfig` - 配置和查看网络接口的状态。
- `netstat` - 打印网络连接，路由表，接口统计信息，伪装连接和组播成员资格。
- `ip` - 显示/操做路由，设备，策略和隧道。
- `ss` - 调查套接字的另一个实用程序。

现在让我们来看看每个命令可以告诉我们什么信息。我们将从 `ifconfig` 开始。

```
user1@vm1:~$ sudo ifconfig

eth0      Link encap:Ethernet  HWaddr 08:00:27:d4:45:68
      # (1), (2), (3)
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.25
5.0   # (4), (5), (6), (7)
          inet6 addr: fe80::a00:27ff:fed4:4568/64 Scope:Link
      # (8), (9), (10)
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
      # (11), (12), (13), (14), (15), (16)
          RX packets:35033 errors:0 dropped:0 overruns:0 frame:0
      # (17), (18), (19), (20), (21), (22)
          TX packets:28590 errors:0 dropped:0 overruns:0 carrier
:0    # (23), (24), (25), (26), (27), (28)
          collisions:0 txqueuelen:1000
      # (29), (30)
          RX bytes:6360747 (6.0 MiB)  TX bytes:21721365 (20.7 Mi
B)   # (31), (32)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:8 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:560 (560.0 B)  TX bytes:560 (560.0 B)
```

我们可以看到 `vm1` 中有两个网络接口， `eth0` 和 `lo` 。 `lo` 是一个回送 接口，用于连接同一台机器上的客户端-服务器程序。

让我们看看我们在 `eth0` 上有哪些信息，它是一个 VirtualBox 的伪网络接口：

字段	描述	字段	描述
(1) Link	物理选项	(17) RX	接收（缩写）
(2) encap	封装类型	(18) packets	封包总数
(3) Hwaddr	MAC 地址	(19) errors	错误的数据包总数
(4) inet	地址族（IPv4）	(20) dropped	丢弃的封包（低系统内存？）
(5) addr	IPv4 地址	(21) overruns	比处理速度快的封包
(6) Bcast	广播地址	(22) frame	接收的无效的帧
(7) Mask	网络掩码	(23) TX	传输（缩写）
(8) inet6	地址族（IPv6）	(24) packets	封包总数
(9) addr	IPv6 地址	(25) errors	错误的数据包总数
(10) Scope	地址范围（主机，链路，全局）	(26) dropped	丢弃的封包（低系统内存？）
(11) UP	接口功能正常	(27) overruns	比处理速度快的封包（我不确定）
(12) BROADCAST	它可以一次性向所有主机发送流量	(28) carrier	链路载波丢失
(13) RUNNING	它准备好接受数据（我不确定）	(29) collisions	发生了包装碰撞
(14) MULTICAST	它可以发送和接收组播封包	(30) txqueuelen	传出数据包的转发队列长度
(15) MTU	其最大传输单位	(31) RX bytes	收到的字节总数
(16) Metric	路由开销（在 Linux 中未使用）	(32) TX bytes	发送的字节总数

这确实很多。但是同样，现在的重要字段是：

- (5) `addr` - IPv4地址。
- (6) `Bcast` - 广播地址。
- (7) `Mask` - 网络掩码。
- (11) `UP` - 接口正常工作。
- (13) `RUNNING` - 准备好接受数据。
- (19) `errors` 和 (25) `errors` - 如果在这里有不为零的东西，我们就有问题了。

现在让我们看看 `netstat` 能给我们看的东西。

```
user1@vm1:~$ sudo netstat -ap
Active Internet connections (servers and established)
#(1)  (2)    (3)    (4)                     (5)
    (6)         (7)
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address
    State       PID/Program name
tcp        0      0 *:sunrpc                *:*
    LISTEN      580/portmap
tcp        0      0 *:ssh                   *:*
    LISTEN      900/sshd
tcp        0      0 localhost:smtp          *:*
    LISTEN      1111/exim4
tcp        0      0 *:36286                 *:*
    LISTEN      610/rpc.statd
tcp        0      0 10.0.2.15:ssh           10.0.2.2:52191
    ESTABLISHED 12023/sshd: user1 [
tcp        0      0 10.0.2.15:ssh           10.0.2.2:48663
    ESTABLISHED 11792/sshd: user1 [
tcp6       0      0 [::]:ssh                [::]:*
    LISTEN      900/sshd
tcp6       0      0 ip6-localhost:smtp      [::]:*
    LISTEN      1111/exim4
udp        0      0 *:bootpc                *:*
                843/dhclient
udp        0      0 *:sunrpc                *:*
                580/portmap
udp        0      0 *:52104                 *:*
                610/rpc.statd
udp        0      0 *:786                   *:*
                610/rpc.statd
#(8)  (9)    (10)        (11)       (12)          (13)     (14)
               (15)
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node   PID/P
rogram name    Path
unix  2      [ ACC ]     STREAM     LISTENING     3452     786/a
cpid           /var/run/acpid.socket
unix  6      [ ]         DGRAM                    3407     751/r
syslogd        /dev/log
unix  2      [ ]         DGRAM                    1940     214/u
devd           @/org/kernel/udev/udevd
unix  2      [ ]         DGRAM                    88528    30939
/sudo
unix  3      [ ]         STREAM     CONNECTED     68565    12023
/sshd: user1 [
unix  3      [ ]         STREAM     CONNECTED     68564    12026
/1
unix  2      [ ]         DGRAM                    68563    12023
/sshd: user1 [
```

```
unix  3      [ ]         STREAM     CONNECTED     66682    11792
/sshd: user1 [
unix  3      [ ]         STREAM     CONNECTED     66681    11794
/0
unix  2      [ ]         DGRAM                    66680    11792
/sshd: user1 [
unix  2      [ ]         DGRAM                    3465     843/d
hclient
unix  2      [ ]         DGRAM                    3448     786/a
cpid
unix  3      [ ]         DGRAM                    1945     214/u
devd
unix  3      [ ]         DGRAM                    1944     214/u
devd
```

我使用两个参数来修改 `netstat` 输出。 `-a` 参数告诉 `netstat` 来显示所有的连接，包括建立的，例如你当前正在打字的 `ssh` 会话，以及监听的，例如等待新的连接的 `sshd` 守护进程。 `-p` 告诉 `netstat` 来显示哪个程序拥有每个连接。

活动互联网连接（服务器和已建立）	
字段	描述
(1) Proto	套接字使用的协议（tcp，udp，raw）
(2) Recv-Q	连接到此套接字的用户程序的未复制的字节数
(3) Send-Q	远程主机未确认的字节数
(4) Local Address	套接字的本端的地址和端口号。
(5) Foreign Address	套接字远端的地址和端口号。
(6) State	`ESTABLISHED`，`SYN_SENT`，`SYN_RECV`，`FIN_WAIT1`，`FIN_WAIT2`，`TIME_WAIT`，`CLOSE`，`CLOSE_WAIT`，`LAST_ACK`，`LISTEN`，`CLOSING`，`UNKNOWN`
(7) PID	拥有套接字的进程的进程 ID（PID）和进程名称的斜杠对。
活动 UNIX 域套接字（服务器和已建立）	
字段	描述
(8) Proto	套接字使用的协议（通常为 unix）。
(9) RefCnt	引用计数（即附加到此套接字的进程）。
(10) Flags	显示的标志是 `SO_ACCEPTON`（显示为 `ACC`），`SO_WAITDATA`（`W`）或 `SO_NOSPACE`（`N`）。
(11) Type	`SOCK_DGRAM`，`SOCK_STREAM`，`SOCK_RAW`，`SOCK_RDM`，`SOCK_SEQPACKET`，`SOCK_PACKET`，`UNKNOWN`。
(12) State	`FREE`，`LISTENING`，`CONNECTING`，`CONNECTED`，`DISCONNECTING`，`(empty)`，`UNKNOWN`。
(13) I-Node	套接字文件的索引节点。
(14) PID	打开套接字的进程的进程 ID（PID）和进程名称。
(15) Path	这是连接到套接字的对应进程的路径名。

并非所有字段都是重要的。通常你只需要查看活动的互联网连接（服务器和已建立）部分，并使用以下字段：

（1） `Proto` - 套接字使用的协议（tcp，udp，raw）。（4） `Local Address` - 套接字本端的地址和端口号。（5） `Foreign Address` - 套接字远端的地址和端口号，仅用于套接字对。（6） `State` - 现在你只应该知道两个状态： `LISTEN` 和 `ESTABLISHED` 。前者意味着你可以连接到这个套接字，第后者的意思是这个套接字已经连接了，在这种情况下， `netstat` 显示你的套接字对。

`ip` 是一个类似于 `ifconfig` 的，具有扩展功能的程序。它来自 `iproute2` 套件，用于在某一天替换 `ifconfig` 。示例输出：

```
user1@vm1:~$ sudo ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKN
OWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
#        (1)        (2)       (3) (4)       (5)       (6)
     (8)       (9)
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_
fast state UP qlen 1000
    link/ether 08:00:27:d4:45:68 brd ff:ff:ff:ff:ff:ff      # (9
),  (10), (11)
    inet 10.0.2.15/24 brd 10.0.2.255 scope global eth0      # (1
2), (13), (14)
    inet6 fe80::a00:27ff:fed4:4568/64 scope link            # (1
5), (16)
       valid_lft forever preferred_lft forever              # (1
7), (18), (19)
```

同样，我们来看看我们在 `eth0` 上有什么信息：

字段	描述
(1) BROADCAST	它可以一次性向所有主机发送流量
(2) MULTICAST	它可以发送和接收组播数据包
(3) UP	它是生效的，逻辑状态
(4) LOWER_UP	驱动器信号 L1 已开启（自 Linux 2.6.17 起）
(5) MTU	最大传输单位
(6) qdisc	排队规则，基本上是流量调度策略
(8) State	物理状态（载体感觉？）
(9) qlen	传出数据包的转发队列长度
(10) link	物理选项
(11) ether	封装类型，MAC 地址
(12) brd	数据链路层（物理）广播地址
(13) inet	IPv4 地址族地址
(14) brd	IPv4 广播
(15) scope	IPv4地址范围（主机，链路，全局）
(16) inet6	IPv6 地址族地址
(17) scope	IPv6地址范围（主机，链路，全局）
(18) valid_lft	IPv6 源地址选择策略
(19) preffered_lft	IPv6 源地址选择策略

你已经知道哪些参数很重要（与 `ifconfig` 相同）。

`ss` 基本上是具有扩展功能的当代 `netstat` 。这是它的示例输出，其解释为留作练习：

```
user1@vm1:~$ sudo ss -ap | cut -c1-200
State      Recv-Q Send-Q    Local Address:Port      Peer Address
:Port
LISTEN     0      128                    *:sunrpc                  *
:*         users:(("portmap",580,5))
LISTEN     0      128                   :::ssh                    ::
:*         users:(("sshd",900,4))
LISTEN     0      128                    *:ssh                     *
:*         users:(("sshd",900,3))
LISTEN     0      20                   ::1:smtp                   ::
:*         users:(("exim4",1111,4))
LISTEN     0      20             127.0.0.1:smtp                    *
:*         users:(("exim4",1111,3))
LISTEN     0      128                    *:36286                   *
:*         users:(("rpc.statd",610,7))
ESTAB      0      0              10.0.2.15:ssh              10.0.2.2
:52191     users:(("sshd",12023,3),("sshd",12026,3))
ESTAB      0      0              10.0.2.15:ssh              10.0.2.2
:48663     users:(("sshd",11792,3),("sshd",11794,3))
```

这用于处理接口，连接和接口地址。但是网络路由呢？你也可以使用几个命令获取它们的信息：

```
user1@vm1:~$ sudo route -n
Kernel IP routing table
# (1)          (2)             (3)             (4)   (5)    (6)
    (7) (8)
Destination    Gateway         Genmask         Flags Metric Ref
    Use Iface
10.0.2.0       *               255.255.255.0   U     0      0
      0 eth0
default        10.0.2.2        0.0.0.0         UG    0      0
      0 eth0
user1@vm1:~$ sudo netstat -nr
Kernel IP routing table #
    (9)
Destination    Gateway         Genmask         Flags   MSS Wind
ow  irtt Iface
10.0.2.0       0.0.0.0         255.255.255.0   U         0 0
       0 eth0
0.0.0.0        10.0.2.2        0.0.0.0         UG        0 0
       0 eth0
user1@vm1:~$ sudo ip route show
# (10)      (11)      (12)          (13)        (14)
10.0.2.0/24 dev eth0  proto kernel  scope link  src 10.0.2.15
#(15)                (16)
default via 10.0.2.2 dev eth0
```

让我们再一次看看字段：

字段	描述
(1) Destination	目标网络或目标主机。
(2) Gateway	网关地址或 `*` ，如果没有设置的话。
(3) Genmask	目标网络的掩码； `255.255.255.255` 为主机目标， `0.0.0.0` 为默认路由。
(4) Flags	Up, Host, Gateway, Reinstate, Dynamically installed, Modified, Addrconf, Cache entry, ! reject.
(5) Metric	目标的“距离”（通常以跳数计算）。最近的内核不使用它，但路由守护进程可能需要它。
(6) Ref	这个路由的引用次数（在 Linux 内核中未使用）。
(7) Use	路由查询次数。
(8) Iface	用于这个路由的，发送封包的接口
(9) irtt	初始 RTT（往返时间）。内核使用它来猜测最佳的 TCP 协议参数，而无需等待（可能很慢）的答案。
(10) Net/Mask	目标网络或目标主机。
(11) dev	用于这个路由的，发送封包的接口
(12) proto	`man ip /RTPROTO` : redirect, kernel, boot, static, ra
(13) scope	`man ip /SCOPE_WALUE` : global, site, link, host
(14) src	发送到路由前缀覆盖的目标时，优先选择的源地址。
(15) default	默认网关地址
(15) dev	用于这个路由的，发送封包的接口

目前的重要字段：

- （1） `Destination` - 目标网络或目标主机。
- （2） `Gateway` - 网关地址或 `*` ，如果没有设置的话。默认值意味着，如果没有明确指定的目标地址的网关，则将通过该网关发送数据包。
- （3） `Genmask` - 目标网络的网络掩码； `255.255.255.255` 为主机目标， `0.0.0.0` 为默认路由。
- （8） `Iface` - 用于这个路由的，发送封包的接口。

`netstat` 和 `route` 的哪个字段对应于`ip route show ·的哪个字段 ，再次留作一个练习。这真是太多了！深吸一口气，让我们转到实践。

现在你将学习如何创建伪接口，为其分配地址并更改其状态。

这样做

```
sudo aptitude install uml-utilities
sudo tunctl -t tap0 -u user1
ls -al /sys/devices/virtual/net/tap0
sudo ifconfig tap0 10.1.1.1 netmask 255.255.255.0
sudo ifconfig
sudo route
ping 10.1.1.1 -c 2
sudo ifconfig tap0 down
ping 10.1.1.1 -c 2
sudo ifconfig tap0 up
sudo ip a a 10.2.2.2/24 brd + dev tap0
sudo ifconfig
sudo route
ip a s
ip r s
ping 10.2.2.2 -c 2
sudo ip link set dev tap0 down
ip a s
ip r s
ping 10.2.2.2 -c 2
sudo tunctl -d tap0
ip a s
ls -al /sys/devices/virtual/net/tap0
```

你会看到什么

```
user1@vm1:~$ sudo aptitude install uml-utilities
The following NEW packages will be installed:
  libfuse2{a} uml-utilities
0 packages upgraded, 2 newly installed, 0 to remove and 0 not up
graded.
Need to get 0 B/205 kB of archives. After unpacking 737 kB will
be used.
Do you want to continue? [Y/n/?]
Selecting previously deselected package libfuse2.
(Reading database ... 39616 files and directories currently inst
alled.)
Unpacking libfuse2 (from .../libfuse2_2.8.4-1.1_amd64.deb) ...
Selecting previously deselected package uml-utilities.
Unpacking uml-utilities (from .../uml-utilities_20070815-1.1_amd
64.deb) ...
Processing triggers for man-db ...
Setting up libfuse2 (2.8.4-1.1) ...
```

```
Setting up uml-utilities (20070815-1.1) ...
Starting User-mode networking switch: uml_switch.
user1@vm1:~$ sudo tunctl -t tap0 -u user1
Set 'tap0' persistent and owned by uid 1000
user1@vm1:~$ ls -al /sys/devices/virtual/net/tap0
total 0
drwxr-xr-x 4 root root    0 Jul 11 05:33 .
drwxr-xr-x 4 root root    0 Jul 11 05:33 ..
-r--r--r-- 1 root root 4096 Jul 11 05:33 address
-r--r--r-- 1 root root 4096 Jul 11 05:33 addr_len
-r--r--r-- 1 root root 4096 Jul 11 05:33 broadcast
-r--r--r-- 1 root root 4096 Jul 11 05:33 carrier
-r--r--r-- 1 root root 4096 Jul 11 05:33 dev_id
-r--r--r-- 1 root root 4096 Jul 11 05:33 dormant
-r--r--r-- 1 root root 4096 Jul 11 05:33 duplex
-r--r--r-- 1 root root 4096 Jul 11 05:33 features
-rw-r--r-- 1 root root 4096 Jul 11 05:33 flags
-r--r--r-- 1 root root 4096 Jul 11 05:33 group
-rw-r--r-- 1 root root 4096 Jul 11 05:33 ifalias
-r--r--r-- 1 root root 4096 Jul 11 05:33 ifindex
-r--r--r-- 1 root root 4096 Jul 11 05:33 iflink
-r--r--r-- 1 root root 4096 Jul 11 05:33 link_mode
-rw-r--r-- 1 root root 4096 Jul 11 05:33 mtu
-r--r--r-- 1 root root 4096 Jul 11 05:33 operstate
-r--r--r-- 1 root root 4096 Jul 11 05:33 owner
drwxr-xr-x 2 root root    0 Jul 11 05:33 power
-r--r--r-- 1 root root 4096 Jul 11 05:33 speed
drwxr-xr-x 2 root root    0 Jul 11 05:33 statistics
lrwxrwxrwx 1 root root    0 Jul 11 05:33 subsystem -> ../../../.
./class/net
-r--r--r-- 1 root root 4096 Jul 11 05:33 tun_flags
-rw-r--r-- 1 root root 4096 Jul 11 05:33 tx_queue_len
-r--r--r-- 1 root root 4096 Jul 11 05:33 type
-rw-r--r-- 1 root root 4096 Jul 11 05:33 uevent
user1@vm1:~$ sudo ifconfig tap0 10.1.1.1 netmask 255.255.255.0
user1@vm1:~$ sudo ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:d4:45:68
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.25
5.0
          inet6 addr: fe80::a00:27ff:fed4:4568/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:64040 errors:0 dropped:0 overruns:0 frame:0
          TX packets:44578 errors:0 dropped:0 overruns:0 carrier
:0
          collisions:0 txqueuelen:1000
          RX bytes:19663646 (18.7 MiB)  TX bytes:25043918 (23.8
MiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:76 errors:0 dropped:0 overruns:0 frame:0
```

```
          TX packets:76 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:6272 (6.1 KiB)  TX bytes:6272 (6.1 KiB)

tap0      Link encap:Ethernet  HWaddr ee:d8:2e:f6:bc:f1
          inet addr:10.1.1.1  Bcast:10.1.1.255  Mask:255.255.255
.0
          inet6 addr: fe80::ecd8:2eff:fef6:bcf1/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:1 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

user1@vm1:~$ sudo route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref
    Use Iface
10.0.2.0        *               255.255.255.0   U     0      0
      0 eth0
10.1.1.0        *               255.255.255.0   U     0      0
      0 tap0
default         10.0.2.2        0.0.0.0         UG    0      0
      0 eth0
user1@vm1:~$ ping 10.1.1.1 -c 2
PING 10.1.1.1 (10.1.1.1) 56(84) bytes of data.
64 bytes from 10.1.1.1: icmp_req=1 ttl=64 time=0.070 ms
64 bytes from 10.1.1.1: icmp_req=2 ttl=64 time=0.027 ms

--- 10.1.1.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.027/0.048/0.070/0.022 ms
user1@vm1:~$ sudo ifconfig tap0 down
user1@vm1:~$ ping 10.1.1.1 -c 2
PING 10.1.1.1 (10.1.1.1) 56(84) bytes of data.
64 bytes from 10.1.1.1: icmp_req=1 ttl=64 time=0.030 ms
64 bytes from 10.1.1.1: icmp_req=2 ttl=64 time=0.024 ms

--- 10.1.1.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.024/0.027/0.030/0.003 ms
user1@vm1:~$ sudo ifconfig tap0 up
user1@vm1:~$ sudo ip a a 10.2.2.2/24 brd + dev tap0
user1@vm1:~$ sudo ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:d4:45:68
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.25
5.0
          inet6 addr: fe80::a00:27ff:fed4:4568/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:64088 errors:0 dropped:0 overruns:0 frame:0
          TX packets:44609 errors:0 dropped:0 overruns:0 carrier
:0
          collisions:0 txqueuelen:1000
```

```
          RX bytes:19667480 (18.7 MiB)  TX bytes:25049771 (23.8
MiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:84 errors:0 dropped:0 overruns:0 frame:0
          TX packets:84 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:6944 (6.7 KiB)  TX bytes:6944 (6.7 KiB)

tap0      Link encap:Ethernet  HWaddr ee:d8:2e:f6:bc:f1
          inet addr:10.1.1.1  Bcast:10.1.1.255  Mask:255.255.255
.0
          inet6 addr: fe80::ecd8:2eff:fef6:bcf1/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:9 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
user1@vm1:~$ sudo route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref
    Use Iface
10.2.2.0        *               255.255.255.0   U     0      0
      0 tap0
10.0.2.0        *               255.255.255.0   U     0      0
      0 eth0
10.1.1.0        *               255.255.255.0   U     0      0
      0 tap0
default         10.0.2.2        0.0.0.0         UG    0      0
      0 eth0
user1@vm1:~$ ip a s
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKN
OWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_
fast state UP qlen 1000
    link/ether 08:00:27:d4:45:68 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global eth0
    inet6 fe80::a00:27ff:fed4:4568/64 scope link
       valid_lft forever preferred_lft forever
12: tap0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo
_fast state UNKNOWN qlen 500
    link/ether ee:d8:2e:f6:bc:f1 brd ff:ff:ff:ff:ff:ff
    inet 10.1.1.1/24 brd 10.1.1.255 scope global tap0
    inet 10.2.2.2/24 brd 10.2.2.255 scope global tap0
    inet6 fe80::ecd8:2eff:fef6:bcf1/64 scope link
       valid_lft forever preferred_lft forever
```

```
user1@vm1:~$ ip r s
10.2.2.0/24 dev tap0  proto kernel  scope link  src 10.2.2.2
10.0.2.0/24 dev eth0  proto kernel  scope link  src 10.0.2.15
10.1.1.0/24 dev tap0  proto kernel  scope link  src 10.1.1.1
default via 10.0.2.2 dev eth0
user1@vm1:~$ ping 10.2.2.2 -c 2
PING 10.2.2.2 (10.2.2.2) 56(84) bytes of data.
64 bytes from 10.2.2.2: icmp_req=1 ttl=64 time=0.081 ms
64 bytes from 10.2.2.2: icmp_req=2 ttl=64 time=0.025 ms

--- 10.2.2.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.025/0.053/0.081/0.028 ms
user1@vm1:~$ sudo ip link set dev tap0 down
user1@vm1:~$ ip a s
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKN
OWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_
fast state UP qlen 1000
    link/ether 08:00:27:d4:45:68 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global eth0
    inet6 fe80::a00:27ff:fed4:4568/64 scope link
       valid_lft forever preferred_lft forever
12: tap0: <BROADCAST,MULTICAST> mtu 1500 qdisc pfifo_fast state
DOWN qlen 500
    link/ether ee:d8:2e:f6:bc:f1 brd ff:ff:ff:ff:ff:ff
    inet 10.1.1.1/24 brd 10.1.1.255 scope global tap0
    inet 10.2.2.2/24 brd 10.2.2.255 scope global tap0
user1@vm1:~$ ip r s
10.0.2.0/24 dev eth0  proto kernel  scope link  src 10.0.2.15
default via 10.0.2.2 dev eth0
user1@vm1:~$ ping 10.2.2.2 -c 2
PING 10.2.2.2 (10.2.2.2) 56(84) bytes of data.
64 bytes from 10.2.2.2: icmp_req=1 ttl=64 time=0.037 ms
64 bytes from 10.2.2.2: icmp_req=2 ttl=64 time=0.024 ms

--- 10.2.2.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.024/0.030/0.037/0.008 ms
user1@vm1:~$ sudo tunctl -d tap0
Set 'tap0' nonpersistent
user1@vm1:~$ ip a s
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKN
OWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_
```

```
fast state UP qlen 1000
    link/ether 08:00:27:d4:45:68 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global eth0
    inet6 fe80::a00:27ff:fed4:4568/64 scope link
       valid_lft forever preferred_lft forever
user1@vm1:~$ ls -al /sys/devices/virtual/net/tap0
ls: cannot access /sys/devices/virtual/net/tap0: No such file or
 directory
user1@vm1:~$
```

解释

1. 安装使用伪（虚拟）接口的软件包。
2. 创建伪接口 `tap0` 。
3. 打印为此接口创建的，虚拟目录的内容，其中包含其设置和统计信息。
4. 将 IP 地址 `10.1.1.1/24` 添加到 `tap0` 。
5. 打印当前接口状态。
6. 打印当前路由表条目。请注意，Linux 自动为 `tap0` 添加新路由。
7. 通过向其发送 ICMP 回显请求数据包来测试 `tap0` 。
8. 将 `tap0` 设为 `DOWN` 状态（停用）。
9. 通过再次发送 ICMP 回显请求数据包来测试 `tap0` 。会有额外的附加题来解释为什么这个仍然可以工作，尽管已经停用了。
10. 将 `tap0` 设为 `UP` 状态（启用）。
11. 向 `tap0` 添加额外的IP地址 `10.2.2.2/24` 。 `ip aa` 是 `ip addr add` 的缩写版本。这个 `+` 的含义，你会在附加题中自己发现它。
12. 打印当前接口状态。注意 `ifconfig` 无法列出使用 `ip` 工具添加的新 IP 地址。为什么？留作附加题。
13. 打印当前路由表。请注意，Linux 自动为 `tap0` 添加了一条路由。
14. 使用 `ip` 工具打印当前接口状态。你可以在这里看到新添加的地址。
15. 使用 `ip` 工具打印我们的路由表。
16. 通过向其发送 ICMP 回显请求报文，来测试 `net tap0` 的 IP 地址。
17. 将 `tap0` 设为 `DOWN` 状态。
18. 打印当前接口状态。
19. 打印当前路由表条目。请注意， `tap0` 路由将自动删除。
20. 通过向其发送 ICMP 回显请求报文，来测试 `net tap0` 的 IP 地址。这个有用，为什么？
21. 删除伪接口 `tap0` 。
22. 打印当前接口状态。 `tap0` 不存在
23. 告诉我们， `tap0` 虚拟目录现在也没有了。

附加题

- 熟悉 `man ifconfig` ， `man ip` ， `man netstat` ， `man ss` 。
- 当 `tap0` 处于关闭状态时，为什么 `ping` 有用？
- `brd +` 的意思是什么？

- 为什么 `ifconfig` 无法列出使用 `ip` 添加的新地址？

练习 25：网络：配置文件，`/etc/network/interfaces`

> 原文：Exercise 25. Networking: configuration files, /etc/network/interfaces
>
> 译者：飞龙
>
>
>
> 自豪地采用谷歌翻译

从命令行配置网络接口是很好的，但现在是时候学习如何让 `vm1` 自动配置网络接口。为此，你将了解 `/etc/network/interfaces` 配置文件：

```
user1@vm1:~$ cat /etc/network/interfaces
# This file describes the network interfaces available on your s
ystem
# and how to activate them. For more information, see interfaces
(5).

# The loopback network interface
#(1) (2)
auto lo
#(3)   (4)(5)  (6)
iface lo inet loopback

# The primary network interface
#(7)           (8)
allow-hotplug eth0
#(9)   (10) (11) (12)
iface eth0 inet dhcp
```

像往常一样，字段及其描述：

字段	描述
（1）	自动配置界面。
（2）	接口名称。
（3）	接口配置的开始
（4）	要配置的接口名称
（5）	此接口使用 TCP/IP 网络，IPv4。
（6）	它是回送接口。默认回送地址将自动分配给它。
（7）	在可用时自动配置接口（请在这里考虑 usb-modem）。
（8）	接口名称。
（9）	接口配置的开始
（10）	要配置的接口名称
（11）	此接口使用 TCP/IP 网络，IPv4。
（12）	此接口通过 DHCP 自动获取其参数。

其他包含网络配置的重要文件，但我们在这里不会碰到他们：

- `/etc/hosts` - 操作系统中使用的计算机文件，用于将主机名映射到 IP 地址。 `hosts` 文件是一个纯文本文件，通常按照惯例命名为 `hosts` 。
- `/etc/hostname` - 分配给连接到计算机网络的设备的标签，并用于识别各种形式的电子通信设备。
- `/etc/resolv.conf` - 各种操作系统中的计算机文件，用于配置域名系统（DNS）解析器库。该文件是纯文本文件，通常由网络管理员或管理系统配置任务的应用创建。 `resolvconf` 程序是 linux 机器上的这样的程序，它管理 `resolv.conf` 文件。

让我们回忆之前练习的 `tap0` 。如果你重新启动 `vm1` ， 它就会消失。当然，你可以通过重新输入相关命令来启用它，但是让我们想象一下，你需要在重新启动后自动使用该命令。

现在，你将学习如何使用 `/etc/network/interfaces` 文件来配置接口。

这样做

```
1: ip a s
2: sudo vim /etc/network/interfaces
```

现在将这些东西添加到配置文件末尾：

```
auto tap0
iface tap0 inet static
        address 10.1.1.2
        netmask 255.255.255.0
        tunctl_user uml-net

allow-hotplug tap1
iface tap1 inet static
        address 10.1.1.3
        netmask 255.255.255.0
```

现在键入 `:wq<ENTER>` 并继续：

```
sudo /etc/init.d/networking start
ip a s
sudo tunctl -t tap1 -u uml-net
ip a s
```

你会看到什么

```
user1@vm1:~$ ip a s
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKN
OWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_
fast state UP qlen 1000
    link/ether 08:00:27:d4:45:68 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global eth0
    inet6 fe80::a00:27ff:fed4:4568/64 scope link
       valid_lft forever preferred_lft forever
user1@vm1:~$ sudo vim /etc/network/interfaces
# and how to activate them. For more information, see interfaces
(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
allow-hotplug eth0
iface eth0 inet dhcp

auto tap0
iface tap0 inet static
```

```
        address 10.2.2.2
        netmask 255.255.255.0
        tunctl_user uml-net

allow-hotplug tap1
iface tap1 inet static
        address 10.3.3.3
        netmask 255.255.255.0
~
"/etc/network/interfaces" 21L, 457C written               21,1-8
        Bot
user1@vm1:~$ sudo /etc/init.d/networking start
Configuring network interfaces...Set 'tap0' persistent and owned
 by uid 104 done.
user1@vm1:~$ ip a s
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKN
OWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_
fast state UP qlen 1000
    link/ether 08:00:27:d4:45:68 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global eth0
    inet6 fe80::a00:27ff:fed4:4568/64 scope link
       valid_lft forever preferred_lft forever
3: tap0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_
fast state UNKNOWN qlen 500
    link/ether 46:63:30:70:b5:21 brd ff:ff:ff:ff:ff:ff
    inet 10.2.2.2/24 brd 10.2.2.255 scope global tap0
    inet6 fe80::4463:30ff:fe70:b521/64 scope link
       valid_lft forever preferred_lft forever
user1@vm1:~$ sudo tunctl -t tap1 -u uml-net
Set 'tap1' persistent and owned by uid 104
user1@vm1:~$ ip a s
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKN
OWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_
fast state UP qlen 1000
    link/ether 08:00:27:d4:45:68 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global eth0
    inet6 fe80::a00:27ff:fed4:4568/64 scope link
       valid_lft forever preferred_lft forever
3: tap0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_
fast state UNKNOWN qlen 500
    link/ether 46:63:30:70:b5:21 brd ff:ff:ff:ff:ff:ff
    inet 10.2.2.2/24 brd 10.2.2.255 scope global tap0
    inet6 fe80::4463:30ff:fe70:b521/64 scope link
```

```
       valid_lft forever preferred_lft forever
4: tap1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_
fast state UNKNOWN qlen 500
    link/ether 8a:ed:90:33:93:55 brd ff:ff:ff:ff:ff:ff
    inet 10.3.3.3/24 brd 10.3.3.255 scope global tap1
    inet6 fe80::88ed:90ff:fe33:9355/64 scope link
       valid_lft forever preferred_lft forever
user1@vm1:~$
```

解释

1. 打印当前接口配置。
2. 编辑 `/etc/network/interfaces` 。
3. 自动配置 `tap0` 。
4. 为 `tap0` 设置以下 IPv4 静态参数。
5. 将 IP 地址 `10.2.2.2` 添加给 `tap0` 。
6. 为此 IP 地址指定网络掩码、参数“广播”和“网络”自动 从这个网络掩码导出。
7. 指定拥有 `tap0` 接口的用户。
8. 由于可读性的空行。
9. 在 `tap1` 接口出现在系统中时，添加以下参数。
10. 为 `tap1` 设置以下 IPv4 静态参数。
11. 将 IP 地址 `10.3.3.3` 添加给tap1。
12. 为此 IP 地址指定网络掩码。
13. 应用网络配置更改。
14. 打印当前接口配置。你可以看到 `tap0` 被添加到接口列表中。
15. 添加 `tap1` 伪接口。
16. 打印当前接口配置。你可以看到`/etc/network/interfaces中指定的参数自动应用于它。

附加题

- 说明如何导出“网络”和“广播”参数。
- 尝试这个： `ping kitty` 。预期会失败。现在添加一个条目到 `/etc/hosts` ，以便你能够成功执行 `ping` 。

练习 26：网络：封包过滤配置， `iptables`

> 原文：Exercise 26. Networking: packet filter configuration, iptables
>
> 译者：飞龙
>
> 协议：CC BY-NC-SA 4.0
>
> 自豪地采用谷歌翻译

让我以引用维基百科上的 `iptables` 来开始：

> `iptables` 是一个用户态应用程序，允许系统管理员配置由 Linux 内核防火墙（实现为不同的 Netfilter 模块）提供的表，以及它存储的链和规则。不同的内核模块和程序目前用于不同的协议； `iptables` 适用于 IPv4， `ip6tables` 适用于 IPv6， `arptables` 适用于 ARP， `ebtables` 适用于以太网帧。

为了使用它，你必须了解以下概念：

- `LINKTYPE_LINUX_SLL` - `tcpdump` 伪链路层协议。
- 以太网帧头部 - 以太网链路上的数据包称为以太网帧。帧以前缀和起始分隔符开始。接下来，每个以太网帧都有一个头部，其特征为源和目标 MAC 地址。帧的中间部分是载荷数据，包含帧中携带的其他协议（例如，互联网协议）的任何头部。该帧以 32 位循环冗余校验（CRC）结束，用于检测传输中数据的任何损坏。
- IPv4 头部 - IP 封包包括头部部分和数据部分。IPv4 封包头部由 14 个字段组成，其中 13 个是必需的。第十四个字段是可选的，适当地命名为： `options` 。
- TCP 段结构 - 传输控制协议接受来自数据流的数据，将其分割成块，并添加 TCP 头部来创建 TCP 段。TCP 段然后被封装成互联网协议（IP）数据报。TCP 段是“信息封包，TCP 用于与对方交换数据”。

我会提醒你，让你获取一些指南：

- 阅读相应的维基百科文章，直到你至少表面上理解了它（但是深入研究当然更好）。
    - 展开站点左侧的 IP 地址树节点，并通过它来以你的方式实现。
    - 展开 TCP 树节点并执行相同操作。
- 阅读 Linux 网络概念介绍。这本指南很好，因为它甚至承认 互联网是为情欲而生的。

比起 Peter Harrison 的优秀指南，我没办法更好地描述 `iptables` 了。如果你从未使用过它，请先阅读本指南。

但是，我将会将理论付诸实践，并 在数据交换的一个非常简单的情况下，逐步展示出 `iptalbes` 内部的内容。第一件事情是主要概念：

`iptables` - 用于在 Linux 内核中设置，维护和检查 IPv4 包过滤规则表的程序。可以定义几个不同的表。每个表包含多个内置链，并且还可以包含用户定义的链。 `ip6tables` - 用于 IPv6 的相同东西。 链 - 可以匹配一组数据包的规则列表。每个规则规定了，如何处理匹配的数据包。这被称为目标，它可能是相同表中的，用户定义的链的跳转。 目标 - 防火墙规则为封包和目标指定了判别标准。如果数据包不匹配，就会检查的链中的下一个规则；如果它匹配，则下一个规则由目标的值指定，该值可以是用户定义链的名称或特殊值之一：

`ACCEPT` - 让包通过。 `DROP` - 将数据包丢弃。 `QUEUE` - 将数据包传递给用户空间。 `RETURN` - 停止遍历此链，并在上一个（调用）链中的下一个规则处恢复。如果达到了内置链的结尾，或者内置链中的一个带有 `RETURN` 的规则匹配它，链策略指定的目标决定了数据包的命运。

现在让我们看看有什么默认的表和内置的链：

表名	内置链	描述
`raw`		该表主要用于配置与 `NOTRACK` 目标结合的连接跟踪的免除。它以较高的优先级在 `netfilter` 钩子中注册，因此在 `ip_conntrack` 或任何其他 IP 表之前调用。
	PREROUTING	用于经过任何网络接口到达的封包。
	OUTPUT	用于本地进程生成的封包。
`mangle`		该表用于专门的数据包更改。
	PREROUTING	用于在路由之前更改传入的数据包。
	OUTPUT	用于在路由之前更改本地生成的数据包。
	INPUT	用于进入本机的数据包。
	FORWARD	用于经过本机的数据包。
	POSTROUTING	用于当数据包打算出去时，更改它们。
`nat`		当遇到创建新连接的数据包时，将查看此表。
	PREROUTING	用于一旦数据包进来，就更改它们。
	OUTPUT	用于在路由之前更改本地生成的数据包。
	POSTROUTING	用于当数据包打算出去时，更改它们。
`filter`		这是默认表（如果没有传入 `-t` 选项）。
	INPUT	用于发往本地套接字的数据包。
	FORWARD	用于经过本机的数据包。
	OUTPUT	用于本地生成的数据包。

好的，我们准备看看它实际如何运作。我会从我的家用计算机，使用 TCP 协议和 `netcat` 向 `vm1` 发送一个字符串 `Hello world!` ， `netcat` 就像 `cat` 一样，但是通过网络。起步：

1. 我将另一个端口，80，转发到我运行 Linux 的家用 PC，所以我能象这样连接它：

```
(My home PC) --> vm1:80
```

2. 我将这个规则添加到 `iptables` ，来记录 `iptables` 内部的数据包发生了什么。

```
sudo iptables -t raw -A PREROUTING -p tcp -m tcp --dport 80 -j T
RACE
sudo iptables -t raw -A INPUT -p tcp -m tcp --sport 80 -j TRACE
```

这是我的 `vm1` 上的 `iptables` 规则集：

```
root@vm1:/home/user1# for i in raw mangle nat filter ; do echo -
e "\n-----" TABLE: $i '-----' ; iptables -t $i -L ; done

----- TABLE: raw -----
Chain PREROUTING (policy ACCEPT)  target     prot opt source
           destination
                                  TRACE      tcp  --  anywhere
           anywhere            tcp dpt:www
Chain OUTPUT (policy ACCEPT)      target     prot opt source
           destination
                                  TRACE      tcp  --  anywhere
           anywhere            tcp spt:www
----- TABLE: mangle -----
Chain PREROUTING (policy ACCEPT)  target     prot opt source
           destination
Chain INPUT (policy ACCEPT)       target     prot opt source
           destination
Chain FORWARD (policy ACCEPT)     target     prot opt source
           destination
Chain OUTPUT (policy ACCEPT)      target     prot opt source
           destination
Chain POSTROUTING (policy ACCEPT) target     prot opt source
           destination

----- TABLE: nat -----
Chain PREROUTING (policy ACCEPT)  target     prot opt source
           destination
Chain POSTROUTING (policy ACCEPT) target     prot opt source
           destination
Chain OUTPUT (policy ACCEPT)      target     prot opt source
           destination

----- TABLE: filter -----
Chain INPUT (policy ACCEPT)       target     prot opt source
           destination
Chain FORWARD (policy ACCEPT)     target     prot opt source
           destination
Chain OUTPUT (policy ACCEPT)      target     prot opt source
           destination
```

你可以看到，没有其它规则了。另一种查看 `iptables` 规则的方式，是使用 `iptables-save` 工具：

```
root@vm1:/home/user1# iptables-save
# Generated by iptables-save v1.4.8 on Fri Jul 13 08:09:04 2012
#(1)
*mangle
#(2)        (3)     (4)   (5)
:PREROUTING ACCEPT [15662:802240]
:INPUT ACCEPT [15662:802240]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [12756:3671974]
:POSTROUTING ACCEPT [12756:3671974]
COMMIT
# Completed on Fri Jul 13 08:09:04 2012
# Generated by iptables-save v1.4.8 on Fri Jul 13 08:09:04 2012
*nat
:PREROUTING ACCEPT [18:792]
:POSTROUTING ACCEPT [42:2660]
:OUTPUT ACCEPT [42:2660]
COMMIT
# Completed on Fri Jul 13 08:09:04 2012
# Generated by iptables-save v1.4.8 on Fri Jul 13 08:09:04 2012
*raw
:PREROUTING ACCEPT [15854:814892]
:OUTPUT ACCEPT [12855:3682054]
-A PREROUTING -p tcp -m tcp --dport 80 -j TRACE
-A OUTPUT -p tcp -m tcp --sport 80 -j TRACE
COMMIT
# Completed on Fri Jul 13 08:09:04 2012
# Generated by iptables-save v1.4.8 on Fri Jul 13 08:09:04 2012
*filter
:INPUT ACCEPT [35107:2459066]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [26433:10670628]
COMMIT
# Completed on Fri Jul 13 08:09:04 2012
```

`iptables-save` 字段如下所示：

字段	描述
(1)	表名称
(2)	链名称
(3)	链策略
(4)	封包计数
(5)	字节计数

3. 我启动 `nc` 来监听端口 80：

```
nc -l 80
```

4. 我向 `vm1` 发送字符串：

```
echo 'Hello, world!' | nc localhost 80
```

下面的是我的家用 PC 和 `vm1` 之间的交换：

```
08:00:05.655339 IP 10.0.2.2.51534 > 10.0.2.15.80: Flags [S], seq
 4164179969, win 65535, options [mss 1460], length 0
08:00:05.655653 IP 10.0.2.15.80 > 10.0.2.2.51534: Flags [S.], se
q 4149908960, ack 4164179970, win 5840, options [mss 1460], leng
th 0
08:00:05.655773 IP 10.0.2.2.51534 > 10.0.2.15.80: Flags [.], ack
 1, win 65535, length 0
08:00:05.655868 IP 10.0.2.2.51534 > 10.0.2.15.80: Flags [P.], se
q 1:15, ack 1, win 65535, length 14
08:00:05.655978 IP 10.0.2.15.80 > 10.0.2.2.51534: Flags [.], ack
 15, win 5840, length 0
08:00:10.037978 IP 10.0.2.2.51534 > 10.0.2.15.80: Flags [F.], se
q 15, ack 1, win 65535, length 0
08:00:10.038287 IP 10.0.2.15.80 > 10.0.2.2.51534: Flags [F.], se
q 1, ack 16, win 5840, length 0
08:00:10.038993 IP 10.0.2.2.51534 > 10.0.2.15.80: Flags [.], ack
 2, win 65535, length 0
```

让我们回忆，数据如何交换。为此，让我们拆开第一个封包。

```
#                  (13)     (15)    (14)      (16)      (20)
 (17)             (25)
08:00:05.655339 IP 10.0.2.2.51534 > 10.0.2.15.80: Flags [S], seq
 4164179969, win 65535,
#        (8)         (9)
options [mss 1460], length 0
#                 (1)  (2)  (3)  (4)                 (5)
#                 ____ ____ ____ ___________________ ____
        0x0000:  0000 0001 0006 5254 0012 3502 0000 0800  ......
RT..5.....
#                 (6)  (7)  (8)  (9)(10,11)(12)  (13)
#                 ____ ____ ____ ____ /\/\ ____ _________
        0x0010:  4500 002c a006 0000 4006 c2b5 0a00 0202  E..,..
..@.......
#                 (14)      (15) (16) (17)      (18)
#                 _________ ____ ____ _________ __________
        0x0020:  0a00 020f c94e 0050 f834 5801 0000 0000  .....N
.P.4X.....
#               (19,20)(21) (22) (23) (24) (25)
#                 /\/\ ____ ____ ____ ____ ____
        0x0030:  6002 ffff 6641 0000 0204 05b4 0000       `...fA
........
```

我们封包中的字段和描述：

DOD 模型层	OSI 模型层	位于	字段	描述
链路	物理/数据链路	`LINUX_SLL` 头部	(1)	封包类型
			(2)	ARPHRD_ 类型
			(3)	链路层 (MAC) 地址长度
			(4)	链路层 (MAC) 源地址
			(5)	协议类型 (IP)
互联网	网络	IPv4 头部	(6)	版本，互联网头部长度，差分服务代码点，显式拥塞通知.
			(7)	总长度
			(8)	身份，主要用于源 IP 数据报的段的唯一性鉴定
			(9)	标志，段的偏移
			(10)	生存时间 (TTL)
			(11)	协议编号

			(12)	头部校验和
			(13)	源 IP 地址
			(14)	目标 IP 地址
传输	传输	TCP 头部	(15)	源 TCP 端口
			(16)	目标 TCP 端口
			(17)	TCP 初始序列号
			(18)	ACK 编号字段 (空的，由于它是第一个封包)
			(19)	
			(20)	SYN TCP 标志
			(21)	TCP 窗口大小
			(22)	TCP 校验和
			(23)	紧急指针
			(24)	可选字段的开始
			(25)	TCP 最大段大小 (最大传输单元 - 40 字节)

让我们看看 `iptables` 中，这个封包发生了什么：

```
#(1)(2)        (3)    (4)   (5)     (6)  (7) (8)              (
9)               (10)  (11)         (12)
raw:PREROUTING:policy:2     IN=eth0 OUT= MAC=08:00:27:d4:45:68:5
2:54:00:12:35:02:08:00 SRC=10.0.2.2 DST=10.0.2.15
#                           (13)   (14)     (15)      (16)   (17
)     (18)      (19)      (20)
                            LEN=44 TOS=0x00 PREC=0x00 TTL=64 ID=
40966 PROTO=TCP SPT=51534 DPT=80
#                           (21)           (22)  (23)         (2
4)     (25)(26)   (27)
                            SEQ=4164179969 ACK=0 WINDOW=65535 RE
S=0x00 SYN URGP=0 OPT (020405B4)
mangle:PREROUTING:policy:1  IN=eth0 OUT= MAC=08:00:27:d4:45:68:5
2:54:00:12:35:02:08:00 SRC=10.0.2.2 DST=10.0.2.15
                            LEN=44 TOS=0x00 PREC=0x00 TTL=64 ID=
40966 PROTO=TCP SPT=51534 DPT=80
                            SEQ=4164179969 ACK=0 WINDOW=65535 RE
S=0x00 SYN URGP=0 OPT (020405B4)
nat:PREROUTING:policy:1     IN=eth0 OUT= MAC=08:00:27:d4:45:68:5
2:54:00:12:35:02:08:00 SRC=10.0.2.2 DST=10.0.2.15
                            LEN=44 TOS=0x00 PREC=0x00 TTL=64 ID=
40966 PROTO=TCP SPT=51534 DPT=80
                            SEQ=4164179969 ACK=0 WINDOW=65535 RE
S=0x00 SYN URGP=0 OPT (020405B4)
mangle:INPUT:policy:1       IN=eth0 OUT= MAC=08:00:27:d4:45:68:5
2:54:00:12:35:02:08:00 SRC=10.0.2.2 DST=10.0.2.15
                            LEN=44 TOS=0x00 PREC=0x00 TTL=64 ID=
40966 PROTO=TCP SPT=51534 DPT=80
                            SEQ=4164179969 ACK=0 WINDOW=65535 RE
S=0x00 SYN URGP=0 OPT (020405B4)
filter:INPUT:policy:1       IN=eth0 OUT= MAC=08:00:27:d4:45:68:5
2:54:00:12:35:02:08:00 SRC=10.0.2.2 DST=10.0.2.15
                            LEN=44 TOS=0x00 PREC=0x00 TTL=64 ID=
40966 PROTO=TCP SPT=51534 DPT=80
                            SEQ=4164179969 ACK=0 WINDOW=65535 RE
S=0x00 SYN URGP=0 OPT (020405B4)
```

`iptables` 日志的字段的描述：

字段	描述
(1)	表名称
(2)	链名称
(3)	类型 (用于内建链的策略)
(4)	规则编号
(5)	输入接口
(6)	输出接口 (空的，因为封包的目标是 vm1 自身)
(7)	MAC 地址
(8)	目标 (vm1) MAC
(9)	源 MAC
(10)	协议类型：IP
(11)	源 IP 地址
(12)	目标 IP 地址
(13)	IP 封包长度，以字节为单位 (不包括链路层头部)
(14)	IP 服务类型
(15)	IP 优先级
(16)	IP 生存时间
(17)	IP 封包 ID
(18)	协议类型：TCP
(19)	TCP 源端口
(20)	TCP 目标端口
(21)	TCP 序列号
(22)	TCP 应答编号
(23)	TCP 窗口大小
(24)	TCP 保留位
(25)	TCP SYN 标志已设置
(25)	TCP 紧急指针未设置
(25)	TCP 选项

现在我将使用 `tcpdump` 输出和 `iptables` 日志，并排（更多的是逐段）向你显示这个交换。你的任务是逐行浏览这个交换，了解会发生什么。我建议你打印这个交换，并使用笔和纸进行处理它，你可以从特殊页面打印它。你需要回答的问题是：

- 每个字段的意思是什么？拿着铅笔，将字段从 `tcpdump` 的踪迹连接到原始数据包的十六进制数据，再到 `iptables` 日志。
- 数据包以什么顺序进行处理？首先是哪个表，最后是哪个，为什么？
- 为什么只有第一个数据包通过 `nat` 表进行处理？

这是输出，看看：

```
08:00:05.655339 IP 10.0.2.2.51534 > 10.0.2.15.80: Flags [S], seq
 4164179969, win 65535, options [mss 1460], length 0
        0x0000:  0000 0001 0006 5254 0012 3502 0000 0800  ......
RT..5.....
        0x0010:  4500 002c a006 0000 4006 c2b5 0a00 0202  E..,..
..@.......
        0x0020:  0a00 020f c94e 0050 f834 5801 0000 0000  .....N
.P.4X.....
        0x0030:  6002 ffff 6641 0000 0204 05b4 0000       `...fA
........
`
raw:PREROUTING:policy:2     IN=eth0 OUT= MAC=08:00:27:d4:45:68:5
2:54:00:12:35:02:08:00 SRC=10.0.2.2 DST=10.0.2.15
                            LEN=44 TOS=0x00 PREC=0x00 TTL=64 ID=
40966 PROTO=TCP SPT=51534 DPT=80
                            SEQ=4164179969 ACK=0 WINDOW=65535 RE
S=0x00 SYN URGP=0 OPT (020405B4)
mangle:PREROUTING:policy:1  IN=eth0 OUT= MAC=08:00:27:d4:45:68:5
2:54:00:12:35:02:08:00 SRC=10.0.2.2 DST=10.0.2.15
                            LEN=44 TOS=0x00 PREC=0x00 TTL=64 ID=
40966 PROTO=TCP SPT=51534 DPT=80
                            SEQ=4164179969 ACK=0 WINDOW=65535 RE
S=0x00 SYN URGP=0 OPT (020405B4)
nat:PREROUTING:policy:1     IN=eth0 OUT= MAC=08:00:27:d4:45:68:5
2:54:00:12:35:02:08:00 SRC=10.0.2.2 DST=10.0.2.15
                            LEN=44 TOS=0x00 PREC=0x00 TTL=64 ID=
40966 PROTO=TCP SPT=51534 DPT=80
                            SEQ=4164179969 ACK=0 WINDOW=65535 RE
S=0x00 SYN URGP=0 OPT (020405B4)
mangle:INPUT:policy:1       IN=eth0 OUT= MAC=08:00:27:d4:45:68:5
2:54:00:12:35:02:08:00 SRC=10.0.2.2 DST=10.0.2.15
                            LEN=44 TOS=0x00 PREC=0x00 TTL=64 ID=
40966 PROTO=TCP SPT=51534 DPT=80
                            SEQ=4164179969 ACK=0 WINDOW=65535 RE
S=0x00 SYN URGP=0 OPT (020405B4)
filter:INPUT:policy:1       IN=eth0 OUT= MAC=08:00:27:d4:45:68:5
2:54:00:12:35:02:08:00 SRC=10.0.2.2 DST=10.0.2.15
                            LEN=44 TOS=0x00 PREC=0x00 TTL=64 ID=
40966 PROTO=TCP SPT=51534 DPT=80
                            SEQ=4164179969 ACK=0 WINDOW=65535 RE
S=0x00 SYN URGP=0 OPT (020405B4)
```

```
08:00:05.655653 IP 10.0.2.15.80 > 10.0.2.2.51534: Flags [S.], se
q 4149908960, ack 4164179970, win 5840, options [mss 1460], leng
th 0
        0x0000:  0004 0001 0006 0800 27d4 4568 0000 0800  ......
..'.Eh....
        0x0010:  4500 002c 0000 4000 4006 22bc 0a00 020f  E..,..
@.@.".....
        0x0020:  0a00 0202 0050 c94e f75a 95e0 f834 5802  .....P
.N.Z...4X.
        0x0030:  6012 16d0 c224 0000 0204 05b4            `....$
......
'
raw:OUTPUT:policy:2         IN= OUT=eth0 SRC=10.0.2.15 DST=10.0.
2.2
                            LEN=44 TOS=0x00 PREC=0x00 TTL=64 ID=
0 DF PROTO=TCP SPT=80 DPT=51534
                            SEQ=4149908960 ACK=4164179970 WINDOW
=5840 RES=0x00 ACK SYN URGP=0 OPT (020405B4) UID=0 GID=0
mangle:OUTPUT:policy:1      IN= OUT=eth0 SRC=10.0.2.15 DST=10.0.
2.2
                            LEN=44 TOS=0x00 PREC=0x00 TTL=64 ID=
0 DF PROTO=TCP SPT=80 DPT=51534
                            SEQ=4149908960 ACK=4164179970 WINDOW
=5840 RES=0x00 ACK SYN URGP=0 OPT (020405B4) UID=0 GID=0
filter:OUTPUT:policy:1      IN= OUT=eth0 SRC=10.0.2.15 DST=10.0.
2.2
                            LEN=44 TOS=0x00 PREC=0x00 TTL=64 ID=
0 DF PROTO=TCP SPT=80 DPT=51534
                            SEQ=4149908960 ACK=4164179970 WINDOW
=5840 RES=0x00 ACK SYN URGP=0 OPT (020405B4) UID=0 GID=0
mangle:POSTROUTING:policy:1 IN= OUT=eth0 SRC=10.0.2.15 DST=10.0.
2.2
                            LEN=44 TOS=0x00 PREC=0x00 TTL=64 ID=
0 DF PROTO=TCP SPT=80 DPT=51534
                            SEQ=4149908960 ACK=4164179970 WINDOW
=5840 RES=0x00 ACK SYN URGP=0 OPT (020405B4) UID=0 GID=0

08:00:05.655773 IP 10.0.2.2.51534 > 10.0.2.15.80: Flags [.], ack
 1, win 65535, length 0
        0x0000:  0000 0001 0006 5254 0012 3502 0000 0800  ......
RT..5.....
        0x0010:  4500 0028 a007 0000 4006 c2b8 0a00 0202  E..(..
..@.......
        0x0020:  0a00 020f c94e 0050 f834 5802 f75a 95e1  .....N
.P.4X..Z..
        0x0030:  5010 ffff f0b1 0000 0000 0000 0000       P.....
........

raw:PREROUTING:policy:2     IN=eth0 OUT= MAC=08:00:27:d4:45:68:5
2:54:00:12:35:02:08:00 SRC=10.0.2.2 DST=10.0.2.15
                            LEN=40 TOS=0x00 PREC=0x00 TTL=64 ID=
40967 PROTO=TCP SPT=51534 DPT=80
```

```
                            SEQ=4164179970 ACK=4149908961 WINDOW
=65535 RES=0x00 ACK URGP=0
mangle:PREROUTING:policy:1  IN=eth0 OUT= MAC=08:00:27:d4:45:68:5
2:54:00:12:35:02:08:00 SRC=10.0.2.2 DST=10.0.2.15
                            LEN=40 TOS=0x00 PREC=0x00 TTL=64 ID=
40967 PROTO=TCP SPT=51534 DPT=80
                            SEQ=4164179970 ACK=4149908961 WINDOW
=65535 RES=0x00 ACK URGP=0
mangle:INPUT:policy:1       IN=eth0 OUT= MAC=08:00:27:d4:45:68:5
2:54:00:12:35:02:08:00 SRC=10.0.2.2 DST=10.0.2.15
                            LEN=40 TOS=0x00 PREC=0x00 TTL=64 ID=
40967 PROTO=TCP SPT=51534 DPT=80
                            SEQ=4164179970 ACK=4149908961 WINDOW
=65535 RES=0x00 ACK URGP=0
filter:INPUT:policy:1       IN=eth0 OUT= MAC=08:00:27:d4:45:68:5
2:54:00:12:35:02:08:00 SRC=10.0.2.2 DST=10.0.2.15
                            LEN=40 TOS=0x00 PREC=0x00 TTL=64 ID=
40967 PROTO=TCP SPT=51534 DPT=80
                            SEQ=4164179970 ACK=4149908961 WINDOW
=65535 RES=0x00 ACK URGP=0

08:00:05.655868 IP 10.0.2.2.51534 > 10.0.2.15.80: Flags [P.], se
q 1:15, ack 1, win 65535, length 14
        0x0000:  0000 0001 0006 5254 0012 3502 0000 0800  ......
RT..5.....
        0x0010:  4500 0036 a008 0000 4006 c2a9 0a00 0202  E..6..
..@.......
        0x0020:  0a00 020f c94e 0050 f834 5802 f75a 95e1  .....N
.P.4X..Z..
        0x0030:  5018 ffff af45 0000 4865 6c6c 6f2c 2077  P....E
..Hello,.w
        0x0040:  6f72 6c64 210a                           orld!.

raw:PREROUTING:policy:2     IN=eth0 OUT= MAC=08:00:27:d4:45:68:5
2:54:00:12:35:02:08:00 SRC=10.0.2.2 DST=10.0.2.15
                            LEN=54 TOS=0x00 PREC=0x00 TTL=64 ID=
40968 PROTO=TCP SPT=51534 DPT=80
                            SEQ=4164179970 ACK=4149908961 WINDOW
=65535 RES=0x00 ACK PSH URGP=0
mangle:PREROUTING:policy:1  IN=eth0 OUT= MAC=08:00:27:d4:45:68:5
2:54:00:12:35:02:08:00 SRC=10.0.2.2 DST=10.0.2.15
                            LEN=54 TOS=0x00 PREC=0x00 TTL=64 ID=
40968 PROTO=TCP SPT=51534 DPT=80
                            SEQ=4164179970 ACK=4149908961 WINDOW
=65535 RES=0x00 ACK PSH URGP=0
mangle:INPUT:policy:1       IN=eth0 OUT= MAC=08:00:27:d4:45:68:5
2:54:00:12:35:02:08:00 SRC=10.0.2.2 DST=10.0.2.15
                            LEN=54 TOS=0x00 PREC=0x00 TTL=64 ID=
40968 PROTO=TCP SPT=51534 DPT=80
                            SEQ=4164179970 ACK=4149908961 WINDOW
=65535 RES=0x00 ACK PSH URGP=0
filter:INPUT:policy:1       IN=eth0 OUT= MAC=08:00:27:d4:45:68:5
2:54:00:12:35:02:08:00 SRC=10.0.2.2 DST=10.0.2.15
```

```
                            LEN=54 TOS=0x00 PREC=0x00 TTL=64 ID=
40968 PROTO=TCP SPT=51534 DPT=80
                            SEQ=4164179970 ACK=4149908961 WINDOW
=65535 RES=0x00 ACK PSH URGP=0

08:00:05.655978 IP 10.0.2.15.80 > 10.0.2.2.51534: Flags [.], ack
 15, win 5840, length 0
        0x0000:  0004 0001 0006 0800 27d4 4568 0000 0800  ......
..'.Eh....
        0x0010:  4500 0028 377c 4000 4006 eb43 0a00 020f  E..(7|
@.@..C....
        0x0020:  0a00 0202 0050 c94e f75a 95e1 f834 5810  .....P
.N.Z...4X.
        0x0030:  5010 16d0 d9d3 0000                      P.....
..
'
raw:OUTPUT:policy:2         IN= OUT=eth0 SRC=10.0.2.15 DST=10.0.
2.2
                            LEN=40 TOS=0x00 PREC=0x00 TTL=64 ID=
14204 DF PROTO=TCP SPT=80 DPT=51534
                            SEQ=4149908961 ACK=4164179984 WINDOW
=5840 RES=0x00 ACK URGP=0
mangle:OUTPUT:policy:1      IN= OUT=eth0 SRC=10.0.2.15 DST=10.0.
2.2
                            LEN=40 TOS=0x00 PREC=0x00 TTL=64 ID=
14204 DF PROTO=TCP SPT=80 DPT=51534
                            SEQ=4149908961 ACK=4164179984 WINDOW
=5840 RES=0x00 ACK URGP=0
filter:OUTPUT:policy:1      IN= OUT=eth0 SRC=10.0.2.15 DST=10.0.
2.2
                            LEN=40 TOS=0x00 PREC=0x00 TTL=64 ID=
14204 DF PROTO=TCP SPT=80 DPT=51534
                            SEQ=4149908961 ACK=4164179984 WINDOW
=5840 RES=0x00 ACK URGP=0
mangle:POSTROUTING:policy:1 IN= OUT=eth0 SRC=10.0.2.15 DST=10.0.
2.2
                            LEN=40 TOS=0x00 PREC=0x00 TTL=64 ID=
14204 DF PROTO=TCP SPT=80 DPT=51534
                            SEQ=4149908961 ACK=4164179984 WINDOW
=5840 RES=0x00 ACK URGP=0

08:00:10.037978 IP 10.0.2.2.51534 > 10.0.2.15.80: Flags [F.], se
q 15, ack 1, win 65535, length 0
        0x0000:  0000 0001 0006 5254 0012 3502 0000 0800  ......
RT..5.....
        0x0010:  4500 0028 a00e 0000 4006 c2b1 0a00 0202  E..(..
..@.......
        0x0020:  0a00 020f c94e 0050 f834 5810 f75a 95e1  .....N
.P.4X..Z..
        0x0030:  5011 ffff f0a2 0000 0000 0000 0000       P.....
........

raw:PREROUTING:policy:2     IN=eth0 OUT= MAC=08:00:27:d4:45:68:5
```

```
2:54:00:12:35:02:08:00 SRC=10.0.2.2 DST=10.0.2.15
                            LEN=40 TOS=0x00 PREC=0x00 TTL=64 ID=
40974 PROTO=TCP SPT=51534 DPT=80
                            SEQ=4164179984 ACK=4149908961 WINDOW
=65535 RES=0x00 ACK FIN URGP=0
mangle:PREROUTING:policy:1  IN=eth0 OUT= MAC=08:00:27:d4:45:68:5
2:54:00:12:35:02:08:00 SRC=10.0.2.2 DST=10.0.2.15
                            LEN=40 TOS=0x00 PREC=0x00 TTL=64 ID=
40974 PROTO=TCP SPT=51534 DPT=80
                            SEQ=4164179984 ACK=4149908961 WINDOW
=65535 RES=0x00 ACK FIN URGP=0
mangle:INPUT:policy:1       IN=eth0 OUT= MAC=08:00:27:d4:45:68:5
2:54:00:12:35:02:08:00 SRC=10.0.2.2 DST=10.0.2.15
                            LEN=40 TOS=0x00 PREC=0x00 TTL=64 ID=
40974 PROTO=TCP SPT=51534 DPT=80
                            SEQ=4164179984 ACK=4149908961 WINDOW
=65535 RES=0x00 ACK FIN URGP=0
filter:INPUT:policy:1       IN=eth0 OUT= MAC=08:00:27:d4:45:68:5
2:54:00:12:35:02:08:00 SRC=10.0.2.2 DST=10.0.2.15
                            LEN=40 TOS=0x00 PREC=0x00 TTL=64 ID=
40974 PROTO=TCP SPT=51534 DPT=80
                            SEQ=4164179984 ACK=4149908961 WINDOW
=65535 RES=0x00 ACK FIN URGP=0

08:00:10.038287 IP 10.0.2.15.80 > 10.0.2.2.51534: Flags [F.], se
q 1, ack 16, win 5840, length 0
        0x0000:  0004 0001 0006 0800 27d4 4568 0000 0800  ......
..'.Eh....
        0x0010:  4500 0028 377d 4000 4006 eb42 0a00 020f  E..(7}
@.@..B....
        0x0020:  0a00 0202 0050 c94e f75a 95e1 f834 5811  .....P
.N.Z...4X.
        0x0030:  5011 16d0 d9d1 0000                      P.....
..
'
raw:OUTPUT:policy:2         IN= OUT=eth0 SRC=10.0.2.15 DST=10.0.
2.2
                            LEN=40 TOS=0x00 PREC=0x00 TTL=64 ID=
14205 DF PROTO=TCP SPT=80 DPT=51534
                            SEQ=4149908961 ACK=4164179985 WINDOW
=5840 RES=0x00 ACK FIN URGP=0 UID=0 GID=0
mangle:OUTPUT:policy:1      IN= OUT=eth0 SRC=10.0.2.15 DST=10.0.
2.2
                            LEN=40 TOS=0x00 PREC=0x00 TTL=64 ID=
14205 DF PROTO=TCP SPT=80 DPT=51534
                            SEQ=4149908961 ACK=4164179985 WINDOW
=5840 RES=0x00 ACK FIN URGP=0 UID=0 GID=0
filter:OUTPUT:policy:1      IN= OUT=eth0 SRC=10.0.2.15 DST=10.0.
2.2
                            LEN=40 TOS=0x00 PREC=0x00 TTL=64 ID=
14205 DF PROTO=TCP SPT=80 DPT=51534
                            SEQ=4149908961 ACK=4164179985 WINDOW
=5840 RES=0x00 ACK FIN URGP=0 UID=0 GID=0
```

```
mangle:POSTROUTING:policy:1 IN= OUT=eth0 SRC=10.0.2.15 DST=10.0.
2.2
                            LEN=40 TOS=0x00 PREC=0x00 TTL=64 ID=
14205 DF PROTO=TCP SPT=80 DPT=51534
                            SEQ=4149908961 ACK=4164179985 WINDOW
=5840 RES=0x00 ACK FIN URGP=0 UID=0 GID=0

08:00:10.038993 IP 10.0.2.2.51534 > 10.0.2.15.80: Flags [.], ack
 2, win 65535, length 0
        0x0000:  0000 0001 0006 5254 0012 3502 0000 0800  ......
RT..5.....
        0x0010:  4500 0028 a00f 0000 4006 c2b0 0a00 0202  E..(..
..@.......
        0x0020:  0a00 020f c94e 0050 f834 5811 f75a 95e2  .....N
.P.4X..Z..
        0x0030:  5010 ffff f0a1 0000 0000 0000 0000       P.....
........

raw:PREROUTING:policy:2     IN=eth0 OUT= MAC=08:00:27:d4:45:68:5
2:54:00:12:35:02:08:00 SRC=10.0.2.2 DST=10.0.2.15
                            LEN=40 TOS=0x00 PREC=0x00 TTL=64 ID=
40975 PROTO=TCP SPT=51534 DPT=80
                            SEQ=4164179985 ACK=4149908962 WINDOW
=65535 RES=0x00 ACK URGP=0
mangle:PREROUTING:policy:1  IN=eth0 OUT= MAC=08:00:27:d4:45:68:5
2:54:00:12:35:02:08:00 SRC=10.0.2.2 DST=10.0.2.15
                            LEN=40 TOS=0x00 PREC=0x00 TTL=64 ID=
40975 PROTO=TCP SPT=51534 DPT=80
                            SEQ=4164179985 ACK=4149908962 WINDOW
=65535 RES=0x00 ACK URGP=0
mangle:INPUT:policy:1       IN=eth0 OUT= MAC=08:00:27:d4:45:68:5
2:54:00:12:35:02:08:00 SRC=10.0.2.2 DST=10.0.2.15
                            LEN=40 TOS=0x00 PREC=0x00 TTL=64 ID=
40975 PROTO=TCP SPT=51534 DPT=80
                            SEQ=4164179985 ACK=4149908962 WINDOW
=65535 RES=0x00 ACK URGP=0
filter:INPUT:policy:1       IN=eth0 OUT= MAC=08:00:27:d4:45:68:5
2:54:00:12:35:02:08:00 SRC=10.0.2.2 DST=10.0.2.15
                            LEN=40 TOS=0x00 PREC=0x00 TTL=64 ID=
40975 PROTO=TCP SPT=51534 DPT=80
                            SEQ=4164179985 ACK=4149908962 WINDOW
=65535 RES=0x00 ACK URGP=0
```

这样做

```
 1: sudo iptables-save
 2: sudo iptables -t filter -A INPUT -i lo -j ACCEPT
 3: sudo iptables -t filter -A INPUT -p tcp -m tcp --dport 22 -j
 ACCEPT
 4: sudo iptables -t filter -P INPUT DROP
 5: sudo iptables -nt filter -L --line-numbers
 6: ping -c 2 -W 1 10.0.2.2
 7: sudo iptables -t filter -A INPUT --match state --state ESTAB
LISHED -j ACCEPT
 8: sudo iptables -nt filter -L --line-numbers
 9: ping -c 2 -W 1 10.0.2.2
10: sudo modprobe ipt_LOG
11: sudo iptables -nt raw -L --line-numbers
12: sudo iptables -t raw -A PREROUTING -p udp -m udp --dport 102
4 -j TRACE
13: sudo iptables -t raw -A OUTPUT -p udp -m udp --sport 1024 -j
 TRACE
14: sudo tail -n0 -f /var/log/kern.log | cut -c52-300 &
15: nc -ulp 1024 &
16: echo 'Hello there!' | nc -u localhost 1000
17: <CTRL+C>
18: fg
19: <CTRL+C>
20: fg
21: <CTRL+C>
```

你会看到什么

```
user1@vm1:~$ sudo iptables-save
# Generated by iptables-save v1.4.8 on Mon Jul 16 09:01:32 2012
*mangle
:PREROUTING ACCEPT [45783:3411367]
:INPUT ACCEPT [45783:3411367]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [30409:9552110]
:POSTROUTING ACCEPT [30331:9543294]
COMMIT
# Completed on Mon Jul 16 09:01:32 2012
# Generated by iptables-save v1.4.8 on Mon Jul 16 09:01:32 2012
*nat
:PREROUTING ACCEPT [24:1056]
:POSTROUTING ACCEPT [755:41247]
:OUTPUT ACCEPT [817:45207]
COMMIT
# Completed on Mon Jul 16 09:01:32 2012
# Generated by iptables-save v1.4.8 on Mon Jul 16 09:01:32 2012
*raw
:PREROUTING ACCEPT [3171:197900]
:OUTPUT ACCEPT [1991:1294054]
```

```
-A PREROUTING -p udp -m udp --dport 80 -j TRACE
-A OUTPUT -p udp -m udp --sport 80 -j TRACE
COMMIT
# Completed on Mon Jul 16 09:01:32 2012
# Generated by iptables-save v1.4.8 on Mon Jul 16 09:01:32 2012
*filter
:INPUT ACCEPT [54:3564]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [28:2540]
COMMIT
# Completed on Mon Jul 16 09:01:32 2012
user1@vm1:~$ sudo iptables -t filter -A INPUT -i lo -j ACCEPT
user1@vm1:~$ sudo iptables -t filter -A INPUT -p tcp -m tcp --dp
ort 22 -j ACCEPT
user1@vm1:~$ sudo iptables -t filter -P INPUT DROP
user1@vm1:~$ sudo iptables -nt filter -L --line-numbers
Chain INPUT (policy DROP)
num  target     prot opt source               destination
1    ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0
2    ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0
  tcp dpt:22

Chain FORWARD (policy ACCEPT)
num  target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
num  target     prot opt source               destination

user1@vm1:~$ ping -c 2 -W 1 10.0.2.2
PING 10.0.2.2 (10.0.2.2) 56(84) bytes of data.

--- 10.0.2.2 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1008ms

user1@vm1:~$ sudo iptables -t filter -A INPUT --match state --st
ate ESTABLISHED -j ACCEPT
user1@vm1:~$ sudo iptables -nt filter -L --line-numbers
Chain INPUT (policy DROP)
num  target     prot opt source               destination
1    ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0
2    ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0
  tcp dpt:22
3    ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0
  state ESTABLISHED

Chain FORWARD (policy ACCEPT)
num  target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
num  target     prot opt source               destination

user1@vm1:~$ ping -c 2 -W 1 10.0.2.2
PING 10.0.2.2 (10.0.2.2) 56(84) bytes of data.
```

```
64 bytes from 10.0.2.2: icmp_req=1 ttl=63 time=0.385 ms
64 bytes from 10.0.2.2: icmp_req=2 ttl=63 time=0.142 ms

--- 10.0.2.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.142/0.263/0.385/0.122 ms
user1@vm1:~$ sudo iptables -nt raw -L --line-numbers
Chain PREROUTING (policy ACCEPT)
num  target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
num  target     prot opt source               destination
user1@vm1:~$ sudo iptables -t raw -A PREROUTING -p udp -m udp --
dport 1024 -j TRACE
user1@vm1:~$ sudo iptables -t raw -A OUTPUT -p udp -m udp --spor
t 1024 -j TRACE
user1@vm1:~$ sudo iptables -nt raw -L --line-numbers
Chain PREROUTING (policy ACCEPT)
num  target     prot opt source               destination
1    TRACE      udp  --  0.0.0.0/0            0.0.0.0/0
  udp dpt:1024

Chain OUTPUT (policy ACCEPT)
num  target     prot opt source               destination
1    TRACE      udp  --  0.0.0.0/0            0.0.0.0/0
  udp spt:1024
user1@vm1:~$ sudo tail -n0 -f /var/log/kern.log | cut -c52-300 &
[1] 10249
user1@vm1:~$ nc -ulp 1024 &
[2] 10251
user1@vm1:~$ echo 'Hello there!' | nc -u localhost 1024
Hello there!
raw:PREROUTING:policy:2 IN=lo OUT= MAC=00:00:00:00:00:00:00:00:0
0:00:00:00:08:00 SRC=127.0.0.1 DST=127.0.0.1 LEN=41 TOS=0x00 PRE
C=0x00 TTL=64 ID=57898 DF PROTO=UDP SPT=50407 DPT=1024 LEN=21
mangle:PREROUTING:policy:1 IN=lo OUT= MAC=00:00:00:00:00:00:00:0
0:00:00:00:00:08:00 SRC=127.0.0.1 DST=127.0.0.1 LEN=41 TOS=0x00
PREC=0x00 TTL=64 ID=57898 DF PROTO=UDP SPT=50407 DPT=1024 LEN=21
mangle:INPUT:policy:1 IN=lo OUT= MAC=00:00:00:00:00:00:00:00:00:
00:00:00:08:00 SRC=127.0.0.1 DST=127.0.0.1 LEN=41 TOS=0x00 PREC=
0x00 TTL=64 ID=57898 DF PROTO=UDP SPT=50407 DPT=1024 LEN=21
filter:INPUT:rule:1 IN=lo OUT= MAC=00:00:00:00:00:00:00:00:00:00
:00:00:08:00 SRC=127.0.0.1 DST=127.0.0.1 LEN=41 TOS=0x00 PREC=0x
00 TTL=64 ID=57898 DF PROTO=UDP SPT=50407 DPT=1024 LEN=21

^C
[2]+  Stopped                 nc -ulp 1024
user1@vm1:~$ fg
nc -ulp 1024
^C
user1@vm1:~$ fg
sudo tail -n0 -f /var/log/kern.log | cut -c52-300
^C
```

```
user1@vm1:~$
```

解释

1. 列出所有表中的所有 `iptables` 规则。你看不到任何东西。
2. 允许 `lo` （环回）接口上的所有传入流量。
3. 允许 TCP 端口 22 上的所有传入流量，这是 `ssh` 。
4. 将默认 `INPUT` 策略更改为 `DROP` ，禁止所有传入连接，除了 TCP 端口 22。如果在此丢失 `vm1` 的连接，则表示你做错了，在 VirtualBox 中重新启动并重试。
5. 列出当前的过滤器规则。注意：你可以按号码删除规则，如下所示： `sudo iptables -t filter -D INPUT 2` 。请注意策略与规则完全不一样。
6. 尝试 `ping` 你的默认网关，失败了。为什么是这样，即使允许传出连接（ `Chain OUTPUT (policy ACCEPT)` ）？传出的数据包被发送到网关，但是网关的回复不能进入。
7. 添加一条规则，告诉 `iptables` 允许属于已建立连接的所有数据包，例如来自 `vm1` 的所有连接。
8. 列出当前的过滤器规则。你可以看到我们的新规则。
9. `ping` `vm1` 的默认网关，这次成功了。
10. 加载 Linux 内核模块，它允许使用包过滤日志功能。
11. 添加规则，来记录所有发往 `vm1` 任何接口的 UDP 端口 1024 的传入数据包。
12. 添加规则，来记录所有来自 `vm1` 任何接口的 UDP 端口 1024 的传出数据包。
13. 列出 `raw` 表规则。
14. 在后台启动 `tail` ，将打印写入 `/var/log/kern.log` 的所有新行。 `cut` 会在开头删除不必要的日志条目前缀。请注意后台进程如何写入终端。
15. 以服务器模式启动 `nc` ，在 `vm1` 的所有接口上监听 UDP 端口 1024 。
16. 以客户端模式启动 `nc` ，将字符串 `Hello there!` 发送到我们的服务器模式 `nc` 。 `tail` 打印出 `kern.log` 中的所有新行，你可以看到在 Linux 内核封包过滤器中，我们的单个 UDP 数据包从一个表到了另一个表。没有回复，所以只有一个数据包被发送和处理。
17. 杀死客户端模式 `nc` 。
18. 将服务器模式 `nc` 带到前台。
19. 杀死服务器模式 `nc` 。
20. 将 `sudo tail -n0 -f /var/log/kern.log | cut -c52-300 &` 带到前台。
21. 杀死它。

附加题

这个练习本身已经很大了。只需要打印出日志，并使用铅笔浏览它，直到理解了每一行的每个字段都发生了什么。如果你卡住了，去问别人：http://nixsrv.com/llthw/ex26/log。

练习 27：安全 Shell，`ssh`，`sshd`，`scp`

> 原文：Exercise 27. Networking: secure shell, ssh, sshd, scp
>
> 译者：飞龙
>
> 协议：CC BY-NC-SA 4.0
>
> 自豪地采用谷歌翻译

你可能已经知道，SSH 是一种网络协议，允许你通过网络登录到 `vm1`。让我们详细研究一下。

> 安全 Shell（SSH）是一种网络协议，用于安全数据通信，远程 Shell 服务或命令执行，以及其它两个联网计算机之间的网络服务，它们通过不安全网络上的安全通道连接：服务器和客户端（运行 SSH 服务器和 SSH 客户端程序）。协议规范区分了两个主要版本，被称为 SSH-1 和 SSH-2。
>
> 协议最著名的应用是，访问类 Unix 操作系统上的 shell 帐户。它为替代 Telnet 和其他不安全的远程 shell 协议而设计，如 Berkeley rsh 和 rexec 协议，它们以明文形式发送信息，特别是密码，使得它们易于使用封包分析来拦截和暴露。SSH 使用的加密 旨在通过不安全的网络（如互联网）提供数据的机密性和完整性。

重要的 SSH 程序，概念和配置文件：

- OpenSSH - 开源的 ssh 程序实现。
- `ssh` - 允许你连接到 SSH 服务器的客户端程序。Putty 就是这样的客户端程序。
- `sshd` - 服务器程序，允许你使用 `ssh` 连接到它。
- `/etc/ssh/ssh_config` - 默认的客户端程序配置文件。
- `/etc/ssh/sshd_config` - 默认服务器程序配置文件。
- 公钥密码系统 - 一种需要两个单独密钥的加密系统，其中一个密钥是私钥，其中一个密钥是公钥。虽然不同，密钥对的两个部分在数学上是相关的。一旦密钥锁定或加密了明文，另一个密钥解锁或解密密文。两个密钥都不能执行这两个功能。其中一个密钥是公开发布的，另一个密钥是保密的。
- SSH 密钥 - SSH 使用公钥密码系统来认证远程计算机，并允许它对用户进行认证（如有必要）。任何人都可以生成一对匹配的不同密钥（公钥和私钥）。公钥放置在所有计算机上，它们允许访问匹配的私钥的所有者（所有者使私钥保密）。虽然认证基于私钥，但认证期间密钥本身不会通过网络传输。
- `/etc/ssh/moduli` - 质数及其生成器，由 `sshd(8)` 用于 Diffie-Hellman Group Exchange 密钥交换方法中。
- `/etc/ssh/ssh_host_dsa_key`，`/etc/ssh/ssh_host_rsa_key` - 主机 RSA 和 DSA 私钥。
- `/etc/ssh/ssh_host_dsa_key.pub`，`/etc/ssh/ssh_host_rsa_key.pub` - 主机 RSA 和 DSA 公钥。

SSH 协议非常重要，因此被广泛使用，并且具有如此多的功能，你必须了解它的一些工作原理。这是它的一些用途：

- `scp` - 通过 SSH 传输文件。
- `sftp` - 类似 ftp 的协议，用于管理远程文件。
- `sshfs` - SSH 上的远程文件系统。
- SSH 隧道 - 一种通过安全连接，传输几乎任何数据的方法。这是非常重要的，因为它可以用于构建受保护系统的基础，以及许多其他用途。

为了了解这个协议，让我们看看，在 SSH 会话中会发生了什么。为此，我们将开始研究 `vm1` 到 `vm1` 的连接的带注解的输出（是的，这是可以做到的，也是完全有效的）。概述：

```
你
    输入 SSH VM1
    控制权现在传递给 SSH 客户端
SSH 客户端
    进入明文阶段
        读取配置
        与 SSH 服务器进行协议协商
    进入 SSH 传输阶段
        与 SSH 服务器进行协商
            数据加密密码
            数据完整性算法
            数据压缩算法
            使用 Diffie-Hellman 算法启动密钥交换
            所得共享密钥用于建立安全连接
    进入 SSH-userauth 阶段
    要求你输入密码
    控制权现在传递给你
你
    输入密码
    控制权现在传递给 SSH 客户端
SSH 客户端
    在 SSH 服务器对你进行认证
    进入 SSH 连接阶段
    为你分配伪终端
    为你启动 shell
    控制权现在传递给你
你
    在 vm1 上做一些（没）有用的事情
    关闭 shell
    控制全现在传递给 SSH 客户端
SSH 客户端
    关闭伪终端
    关闭连接
```

现在阅读这个：

- SSH 协议揭秘
- http://www.cs.ust.hk/faculty/cding/COMP581/SLIDES/slide24.pdf

并研究 SSH 会话的真实输出：

```
user1@vm1:~$ ssh -vv vm1

Protocol version selection, plaintext
-------------------------------------

OpenSSH_5.5p1 Debian-6+squeeze2, OpenSSL 0.9.8o 01 Jun 2010
# Speaks for itself, I will mark such entries with -- below
debug1: Reading configuration data /etc/ssh/ssh_config
# Applying default options for all hosts. Additional options for
 each host may be
# specified in the configuration file
debug1: Applying options for *
debug2: ssh_connect: needpriv 0
debug1: Connecting to vm1 [127.0.1.1] port 22.
debug1: Connection established.
debug1: identity file /home/user1/.ssh/id_rsa type -1      # no
such files
debug1: identity file /home/user1/.ssh/id_rsa-cert type -1
debug1: identity file /home/user1/.ssh/id_dsa type -1
debug1: identity file /home/user1/.ssh/id_dsa-cert type -1
debug1: Remote protocol version 2.0, remote software version Ope
nSSH_5.5p1 Debian-6+squeeze2
debug1: match: OpenSSH_5.5p1 Debian-6+squeeze2 pat OpenSSH*
debug1: Enabling compatibility mode for protocol 2.0
debug1: Local version string SSH-2.0-OpenSSH_5.5p1 Debian-6+sque
eze2
debug2: fd 3 setting O_NONBLOCK

SSH-transport, binary packet protocol
-------------------------------------

debug1: SSH2_MSG_KEXINIT sent
debug1: SSH2_MSG_KEXINIT received
# Key exchange algorithms
debug2: kex_parse_kexinit: diffie-hellman-group-exchange-sha256,
diffie-hellman-group-exchange-sha1,diffie-hellman-group14-sha1,d
iffie-hellman-group1-sha1
# SSH host key types
debug2: kex_parse_kexinit: ssh-rsa-cert-v00@openssh.com,ssh-dss-
cert-v00@openssh.com,ssh-rsa,ssh-dss
# Data encryption ciphers
debug2: kex_parse_kexinit: aes128-ctr,aes192-ctr,aes256-ctr,arcf
our256,arcfour128,aes128-cbc,3des-cbc,blowfish-cbc,cast128-cbc,a
es192-cbc,aes256-cbc,arcfour,rijndael-cbc@lysator.liu.se
debug2: kex_parse_kexinit: aes128-ctr,aes192-ctr,aes256-ctr,arcf
our256,arcfour128,aes128-cbc,3des-cbc,blowfish-cbc,cast128-cbc,a
es192-cbc,aes256-cbc,arcfour,rijndael-cbc@lysator.liu.se
# Data integrity algorithms
debug2: kex_parse_kexinit: hmac-md5,hmac-sha1,umac-64@openssh.co
m,hmac-ripemd160,hmac-ripemd160@openssh.com,hmac-sha1-96,hmac-md
```

```
5-96
debug2: kex_parse_kexinit: hmac-md5,hmac-sha1,umac-64@openssh.co
m,hmac-ripemd160,hmac-ripemd160@openssh.com,hmac-sha1-96,hmac-md
5-96
# Data compression algorithms
debug2: kex_parse_kexinit: none,zlib@openssh.com,zlib
debug2: kex_parse_kexinit: none,zlib@openssh.com,zlib
debug2: kex_parse_kexinit:
debug2: kex_parse_kexinit:
debug2: kex_parse_kexinit: first_kex_follows
debug2: kex_parse_kexinit: reserved 0
# Messages back from server
debug2: kex_parse_kexinit: diffie-hellman-group-exchange-sha256,
diffie-hellman-group-exchange-sha1,diffie-hellman-group14-sha1,d
iffie-hellman-group1-sha1
debug2: kex_parse_kexinit: ssh-rsa,ssh-dss
debug2: kex_parse_kexinit: aes128-ctr,aes192-ctr,aes256-ctr,arcf
our256,arcfour128,aes128-cbc,3des-cbc,blowfish-cbc,cast128-cbc,a
es192-cbc,aes256-cbc,arcfour,rijndael-cbc@lysator.liu.se
debug2: kex_parse_kexinit: aes128-ctr,aes192-ctr,aes256-ctr,arcf
our256,arcfour128,aes128-cbc,3des-cbc,blowfish-cbc,cast128-cbc,a
es192-cbc,aes256-cbc,arcfour,rijndael-cbc@lysator.liu.se
debug2: kex_parse_kexinit: hmac-md5,hmac-sha1,umac-64@openssh.co
m,hmac-ripemd160,hmac-ripemd160@openssh.com,hmac-sha1-96,hmac-md
5-96
debug2: kex_parse_kexinit: hmac-md5,hmac-sha1,umac-64@openssh.co
m,hmac-ripemd160,hmac-ripemd160@openssh.com,hmac-sha1-96,hmac-md
5-96
debug2: kex_parse_kexinit: none,zlib@openssh.com
debug2: kex_parse_kexinit: none,zlib@openssh.com
debug2: kex_parse_kexinit:
debug2: kex_parse_kexinit:
debug2: kex_parse_kexinit: first_kex_follows 0
debug2: kex_parse_kexinit: reserved 0
# Message authentication code setup
debug2: mac_setup: found hmac-md5
debug1: kex: server->client aes128-ctr hmac-md5 none
debug2: mac_setup: found hmac-md5
debug1: kex: client->server aes128-ctr hmac-md5 none
# Key exchange
debug1: SSH2_MSG_KEX_DH_GEX_REQUEST(1024<1024<8192) sent
debug1: SSH2_MSG_KEX_DH_GEX_REQUEST(1024<1024<8192) sent
debug1: expecting SSH2_MSG_KEX_DH_GEX_GROUP
debug2: dh_gen_key: priv key bits set: 135/256
debug2: bits set: 498/1024
debug1: SSH2_MSG_KEX_DH_GEX_INIT sent
debug1: expecting SSH2_MSG_KEX_DH_GEX_REPLY
# Server authentication. vm1 host key is not known because it is
 our first connection
debug2: no key of type 0 for host vm1
debug2: no key of type 2 for host vm1
# Confirmation of host key acceptance
The authenticity of host 'vm1 '(127.0.1.1)' can't be established
```

```
.
RSA key fingerprint is b6:06:92:5e:04:49:d9:e8:57:90:61:1b:16:87
:bb:09.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'vm1' (RSA) to the list of known host
s.
# Key is added to /home/user1/.ssh/known_hosts and checked
debug2: bits set: 499/1024
debug1: ssh_rsa_verify: signature correct
# Based on shared master key, data encryption key and data integ
rity key are derived
debug2: kex_derive_keys
debug2: set_newkeys: mode 1
# Information about this is sent to server
debug1: SSH2_MSG_NEWKEYS sent
debug1: expecting SSH2_MSG_NEWKEYS
debug2: set_newkeys: mode 0
debug1: SSH2_MSG_NEWKEYS received
# IP roaming not enabled? Not sure about this.
debug1: Roaming not allowed by server

SSH-userauth
------------

debug1: SSH2_MSG_SERVICE_REQUEST sent
debug2: service_accept: ssh-userauth
debug1: SSH2_MSG_SERVICE_ACCEPT received
debug2: key: /home/user1/.ssh/id_rsa ((nil))
debug2: key: /home/user1/.ssh/id_dsa ((nil))
debug1: Authentications that can continue: publickey,password
debug1: Next authentication method: publickey
debug1: Trying private key: /home/user1/.ssh/id_rsa
debug1: Trying private key: /home/user1/.ssh/id_dsa
debug2: we did not send a packet, disable method
debug1: Next authentication method: password
user1@vm1''s password:
debug2: we sent a password packet, wait for reply
debug1: Authentication succeeded (password).

SSH-connection
--------------

debug1: channel 0: new [client-session]
debug2: channel 0: send open
# Disable SSH mutiplexing.
# More info: http://www.linuxjournal.com/content/speed-multiple-
ssh-connections-same-server
debug1: Requesting no-more-sessions@openssh.com
debug1: Entering interactive session.
debug2: callback start
debug2: client_session2_setup: id 0
debug2: channel 0: request pty-req confirm 1
# Sending environment variables
```

```
debug1: Sending environment.
debug1: Sending env LANG = en_US.UTF-8
debug2: channel 0: request env confirm 0
debug2: channel 0: request shell confirm 1
# Set TCP_NODELAY flag: http://en.wikipedia.org/wiki/Nagle%27s_a
lgorithm
debug2: fd 3 setting TCP_NODELAY
debug2: callback done
# Connection opened
debug2: channel 0: open confirm rwindow 0 rmax 32768
debug2: channel_input_status_confirm: type 99 id 0
# Pseudo terminal allocation
debug2: PTY allocation request accepted on channel 0
debug2: channel 0: rcvd adjust 2097152
debug2: channel_input_status_confirm: type 99 id 0
# Shell is started
debug2: shell request accepted on channel 0
# Loggin in is completed
Linux vm1 2.6.32-5-amd64 #1 SMP Sun May 6 04:00:17 UTC 2012 x86_
64

The programs included with the Debian GNU/Linux system are free
software;
the exact distribution terms for each program are described in t
he
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the exten
t
permitted by applicable law.
You have mail.
Last login: Thu Jul 19 05:14:40 2012 from 10.0.2.2
user1@vm1:~$ debug2: client_check_window_change: changed
debug2: channel 0: request window-change confirm 0
user1@vm1:~$ debug2: client_check_window_change: changed
debug2: channel 0: request window-change confirm 0
user1@vm1:~$ logout

Ending ssh connection
---------------------

debug2: channel 0: rcvd eof   # end of file
debug2: channel 0: output open -> drain
debug2: channel 0: obuf empty
debug2: channel 0: close_write
debug2: channel 0: output drain -> closed
debug1: client_input_channel_req: channel 0 rtype exit-status re
ply 0
# signalling that channels are half-closed for writing, through
a channel protocol extension
# notification "eow@openssh.com" http://www.openssh.com/txt/rele
ase-5.1
debug1: client_input_channel_req: channel 0 rtype eow@openssh.co
```

```
m reply 0
debug2: channel 0: rcvd eow
# Ending connection
debug2: channel 0: close_read
debug2: channel 0: input open -> closed
debug2: channel 0: rcvd close
debug2: channel 0: almost dead
debug2: channel 0: gc: notify user
debug2: channel 0: gc: user detached
debug2: channel 0: send close
debug2: channel 0: is dead
debug2: channel 0: garbage collecting
debug1: channel 0: free: client-session, nchannels 1
Connection to vm1 closed.
Transferred: sent 1928, received 2632 bytes, in 93.2 seconds
Bytes per second: sent 20.7, received 28.2
debug1: Exit status 0
user1@vm1:~$
```

现在，你将学习如何在调试模式下启动 `sshd`，使用 `scp` 建立公钥认证和复制文件。

这样做

```
 1: mkdir -v ssh_test
 2: cd ssh_test
 3: cp -v /etc/ssh/sshd_config .
 4: sed -i'.bak' 's/^Port 22$/Port 1024/' sshd_config
 5: sed -i 's/^HostKey \/etc\/ssh\/ssh_host_rsa_key$/Hostkey \/h
ome\/user1\/ssh_test\/ssh_host_rsa_key/' sshd_config
 6: sed -i 's/^HostKey \/etc\/ssh\/ssh_host_dsa_key$/Hostkey \/h
ome\/user1\/ssh_test\/ssh_host_dsa_key/' sshd_config
 7: diff sshd_config.bak sshd_config
 8: ssh-keygen -b 4096 -t rsa -N '' -v -h -f ssh_host_rsa_key
 9: ssh-keygen -b 1024 -t dsa -N '' -v -h -f ssh_host_dsa_key
10: ssh-keygen -b 4096 -t rsa -N '' -v  -f ~/.ssh/id_rsa
11: cat ~/.ssh/id_rsa.pub > ~/.ssh/authorized_keys
12: /usr/sbin/sshd -Ddf sshd_config > sshd.out 2>&1 &
13: ssh-keyscan -H vm1 127.0.0.1 >> ~/.ssh/known_hosts
14: /usr/sbin/sshd -Ddf sshd_config >> sshd.out 2>&1 &
15: ssh vm1 -v -p 1024 2>ssh.out
16: ps au --forest
17: logout
18: /usr/sbin/sshd -Ddf sshd_config >> sshd.out 2>&1 &
19: scp -v -P 1024 vm1:.bashrc . 2>scp.out
```

你会看到什么

```
user1@vm1:~$ mkdir -v ssh_test
mkdir: created directory 'ssh_test'
user1@vm1:~$ cd ssh_test
user1@vm1:~/ssh_test$ cp -v /etc/ssh/sshd_config .
'/etc/ssh/sshd_config' -> './sshd_config'
user1@vm1:~/ssh_test$ sed -i'.bak' 's/^Port 22$/Port 1024/' sshd
_config
user1@vm1:~/ssh_test$ sed -i 's/^HostKey \/etc\/ssh\/ssh_host_rs
a_key$/Hostkey \/home\/user1\/ssh_test\/ssh_host_rsa_key/' sshd_
config
user1@vm1:~/ssh_test$ sed -i 's/^HostKey \/etc\/ssh\/ssh_host_ds
a_key$/Hostkey \/home\/user1\/ssh_test\/ssh_host_dsa_key/' sshd_
config
user1@vm1:~/ssh_test$ diff sshd_config.bak sshd_config
5c5
< Port 22
---
> Port 1024
11,12c11,12
< HostKey /etc/ssh/ssh_host_rsa_key
< HostKey /etc/ssh/ssh_host_dsa_key
---
> Hostkey /home/user1/ssh_test/ssh_host_rsa_key
> Hostkey /home/user1/ssh_test/ssh_host_dsa_key
user1@vm1:~/ssh_test$ ssh-keygen -b 4096 -t rsa -N '' -v -h -f s
sh_host_rsa_key
Generating public/private rsa key pair.
Your identification has been saved in ssh_host_rsa_key.
Your public key has been saved in ssh_host_rsa_key.pub.
The key fingerprint is:
8c:0a:8d:ae:c7:34:e6:29:9c:c2:14:29:b8:d9:1d:34 user1@vm1
'The key's randomart image is:
+--[ RSA 4096]----+
|                 |
|    E            |
|. .. .           |
|oo o.   o        |
|.++.... S        |
|oo=...           |
|+=oo.            |
|o==              |
|oo               |
+-----------------+
user1@vm1:~/ssh_test$ ssh-keygen -b 1024 -t dsa -N '' -v -h -f s
sh_host_dsa_key
Generating public/private dsa key pair.
Your identification has been saved in ssh_host_dsa_key.
Your public key has been saved in ssh_host_dsa_key.pub.
The key fingerprint is:
cd:6b:2a:a2:ba:80:65:71:85:ef:2e:6a:c0:a7:d9:aa user1@vm1
'The key's randomart image is:
+--[ DSA 1024]----+
```

```
|     ..          |
|    ..           |
|  . ..           |
|   o  .  o       |
|. o  .  S o      |
|o+ .  .    .     |
|o.=  .    o      |
|.o..o o  o       |
|E=+o o ..        |
+-----------------+
user1@vm1:~/ssh_test$ ssh-keygen -b 4096  -t rsa -N '' -v  -f ~/
.ssh/id_rsa
Generating public/private rsa key pair.
Your identification has been saved in /home/user1/.ssh/id_rsa.
Your public key has been saved in /home/user1/.ssh/id_rsa.pub.
The key fingerprint is:
50:65:18:61:3f:41:36:07:4f:40:36:a7:4b:6d:64:28 user1@vm1
'The key's randomart image is:
+--[ RSA 4096]----+
|        =B&+*    |
|       oE=.&     |
|      .  .= +    |
|        . . +    |
|        S .      |
|                 |
|                 |
|                 |
|                 |
+-----------------+
user1@vm1:~/ssh_test$ cat ~/.ssh/id_rsa.pub > ~/.ssh/authorized_
keys
user1@vm1:~/ssh_test$ /usr/sbin/sshd -Ddf sshd_config > sshd.out
 2>&1 &
[2] 26896
user1@vm1:~/ssh_test$ ssh-keyscan -H vm1 127.0.0.1 >> ~/.ssh/kno
wn_hosts
# 127.0.0.1 SSH-2.0-OpenSSH_5.5p1 Debian-6+squeeze2
# vm1 SSH-2.0-OpenSSH_5.5p1 Debian-6+squeeze2
[2]+  Exit 255                /usr/sbin/sshd -Ddf sshd_config >
sshd.out 2>&1
user1@vm1:~/ssh_test$ /usr/sbin/sshd -Ddf sshd_config >> sshd.ou
t 2>&1 &
[1] 26957
user1@vm1:~/ssh_test$ ssh vm1 -v -p 1024 2>ssh.out
Linux vm1 2.6.32-5-amd64 #1 SMP Sun May 6 04:00:17 UTC 2012 x86_
64

The programs included with the Debian GNU/Linux system are free
software;
the exact distribution terms for each program are described in t
he
individual files in /usr/share/doc/*/copyright.
```

```
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the exten
t
permitted by applicable law.
You have mail.
Last login: Fri Jul 20 09:10:30 2012 from vm1.site
Environment:
  LANG=en_US.UTF-8
  USER=user1
  LOGNAME=user1
  HOME=/home/user1
  PATH=/usr/local/bin:/usr/bin:/bin:/usr/bin/X11:/usr/games
  MAIL=/var/mail/user1
  SHELL=/bin/bash
  SSH_CLIENT=127.0.1.1 47456 1024
  SSH_CONNECTION=127.0.1.1 47456 127.0.1.1 1024
  SSH_TTY=/dev/pts/0
  TERM=xterm
user1@vm1:~$ ps au --forest
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME
 COMMAND
user1    26224  0.0  1.2  23660  6576 pts/2    Ss   09:09   0:01
 -bash
user1    27020  1.0  0.6  68392  3236 pts/2    S    09:50   0:00
  \_ sshd: user1 [priv]
user1    27025  0.0  0.2  68392  1412 pts/2    S    09:50   0:00
  |   \_ sshd: user1@pts/0
user1    27026  9.0  1.2  23564  6404 pts/0    Ss   09:50   0:00
  |       \_ -bash
user1    27051  0.0  0.2  16308  1060 pts/0    R+   09:50   0:00
  |           \_ ps au --forest
user1    27021  1.1  0.5  38504  2880 pts/2    S+   09:50   0:00
  \_ ssh vm1 -v -p 1024
root      1107  0.0  0.1   5932   620 tty6     Ss+  Jul18   0:00
 /sbin/getty 38400 tty6
root      1106  0.0  0.1   5932   616 tty5     Ss+  Jul18   0:00
 /sbin/getty 38400 tty5
root      1105  0.0  0.1   5932   620 tty4     Ss+  Jul18   0:00
 /sbin/getty 38400 tty4
root      1104  0.0  0.1   5932   620 tty3     Ss+  Jul18   0:00
 /sbin/getty 38400 tty3
root      1103  0.0  0.1   5932   616 tty2     Ss+  Jul18   0:00
 /sbin/getty 38400 tty2
root      1102  0.0  0.1   5932   616 tty1     Ss+  Jul18   0:00
 /sbin/getty 38400 tty1
user1@vm1:~$ logout
user1@vm1:~/ssh_test$
[1]+  Exit 255                /usr/sbin/sshd -Ddf sshd_config >
sshd.out 2>&1
user1@vm1:~/ssh_test$ /usr/sbin/sshd -Ddf sshd_config >> sshd.ou
t 2>&1 &
[1] 27067
user1@vm1:~/ssh_test$ scp -v -P 1024 vm1:.bashrc . 2>scp.out
Environment:
```

```
    LANG=en_US.UTF-8
    USER=user1
    LOGNAME=user1
    HOME=/home/user1
    PATH=/usr/local/bin:/usr/bin:/bin:/usr/bin/X11:/usr/games
    MAIL=/var/mail/user1
    SHELL=/bin/bash
    SSH_CLIENT=127.0.1.1 47459 1024
    SSH_CONNECTION=127.0.1.1 47459 127.0.1.1 1024
.bashrc
                                                                  100%
 3184     3.1KB/s   00:00
[1]+  Exit 255                /usr/sbin/sshd -Ddf sshd_config >>
 sshd.out 2>&1
```

解释

1. 创建 `/home/user1/ssh_test` 目录。
2. 使其成为当前工作目录。
3. 将 `sshd_config` 复制到此目录。
4. 将 `sshd` 监听端口从 22 更改为 1024，将副本命名为 `sshd_config.bak` 。
5. 替换 RSA 主机密钥位置。
6. 替换 DSA 主机密钥位置。
7. 显示 `sshd_config` 的旧版本和新版本之间的差异。
8. 生成具有空密码的，新的 4096 位 RSA 主机密钥对，将其保存到 `/home/user1/ssh_test/ssh_host_rsa_key` 和 `/home/user1/ssh_test/` 。
9. 同样的，但是对 DSA 密钥执行。
10. 生成新的认证密钥对，将其保存到 `/home/user1/.ssh/id_rsa` 和 `/home/user1/.ssh/id_rsa.pub` 。
11. 将 `id_rsa.pub` 复制到 `/home/user1/.ssh/authorized_keys` ，来允许无密码认证。
12. 在调试模式下，在端口 1024 上启动新的 SSH 服务器，将所有输出保存到 `sshd.log` 。
13. 提取 SSH 客户端的主机认证密钥，并将其提供给 `/home/user1/.ssh/known_hosts` 。
14. 在调试模式下，在端口 1024 上启动新的 SSH 服务器，将所有输出附加到 `sshd.log` 。这是因为在调试模式下， SSH 服务器只维护一个连接。
15. 使用 `ssh` 客户端连接到此服务器。
16. 以树形式打印当前正在运行的进程。你可以看到，你正在使用 `sshd` 启动的 bash，它服务于你的连接，而 `sshd` 又是由 `sshd` 启动，你在几行之前启动了你自己。。
17. 退出 `ssh` 会话。
18. 再次启动 SSH 服务器。
19. 将文件 `.bashrc` 从你的主目录复制到当前目录。

附加题

观看此视频，它解释了加密如何工作：http://www.youtube.com/watch?v=3QnD2c4Xovk 阅读：http://docstore.mik.ua/orelly/networking_2ndEd/ssh/ch03_04.htm 阅读文件 `ssh.out` ，`scp.out` 和 `sshd.out` 中的调试输出。向你自己解释发生了什么。

练习 28：性能：获取性能情况，`uptime`，`free`，`top`

> 原文：Exercise 28. Performance: getting performance stats, uptime, free, top
>
> 译者：飞龙
>
> 协议：CC BY-NC-SA 4.0
>
> 自豪地采用谷歌翻译

这个练习很简单。首先，我们需要什么样的性能数据？

- CPU 使用情况：
  - 它的负载如何？
  - 哪些进程正在使用它？
- 内存使用情况：
  - 使用了多少内存？
  - 多少内存是空闲的？
  - 多少内存用于缓存？
  - 哪些进程消耗了它？
- 磁盘使用情况：
  - 执行多少输入/输出操作？
  - 由哪个进程？
- 网络使用情况：
  - 传输了多少数据？
  - 由哪个进程？
- 进程情况：
  - 有多少进程？
  - 他们在做什么 工作，还是等待什么？
  - 如果在等待什么，它是什么呢？CPU，磁盘，网络？

为了获取这些情况，我们可以使用以下工具：

- `uptime` - 系统运行了多长时间。
- `free` - 显示系统中可用和使用的内存量。
- `vmstat` - 进程，内存，分页，块 IO，陷阱，磁盘和 cpu 活动的信息。
- `top` - 实时显示 Linux 任务。

我们来看看这个程序及其输出。

`uptime` 的输出：

```
user1@vm1:~$ uptime
#(1)      (2)                  (3)                     (4)    (5)
(6)
 03:13:58 up 4 days, 22:45,  1 user,  load average: 0.00, 0.00,
0.00
```

字段和描述：

字段	描述
（1）	当前时间。
（2）	正常运行时间（启动后的时间）。
（3）	目前有多少用户登录。
（4）	过去 1 分钟的 CPU 负载。这不是规范化的，所以负载均值为 1 意味着单个 CPU 的满负载，但是在 4 个 CPU 的系统上，这意味着它有 75% 空闲时间。
（5）	过去 5 分钟的 CPU 负载。
（6）	过去 15 分钟的 CPU 负载。

`free` 的输出：

```
user1@vm1:~$ free -mt
#             (1)          (2)         (3)          (4)          (5)
      (6)
             total        used        free       shared      buffers
   cached
Mem:           496         267         229            0           27
      196
#                          (7)         (8)
-/+ buffers/cache:          43         453
# 9
Swap:          461           0         461
# 10
Total:         958         267         691
```

字段和描述：

字段	描述
（1）	物理内存总量。
（2）	使用的物理内存总量。
（3）	空闲的物理内存总量。
（4）	共享内存列应该被忽略；它已经过时了。
（5）	专用于缓存磁盘块的 RAM 和文件系统元数据总量。
（6）	专用于从文件读取的页面的 RAM 总量。
（7）	物理内存总量，不包括缓冲区和缓存，（2）-（5）-（6）
（8）	空闲的物理内存总量，包括空闲的缓冲区和缓存，（1）-（7）
（9）	交换文件使用信息。
（10）	总内存使用信息，包括交换内存

`vmstat` 输出：

```
user1@vm1:~$ vmstat -S M
procs -----------memory---------- ---swap-- -----io---- -system-
- ----cpu----
#(1,2)  (3)    (4)    (5)   (6)    (7)   (8)   (9)  (10) (11) (1
2,13,14,15,16)
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   c
s us sy id wa
 0  0      0    229     27    196    0    0     0     0   11
6  0  0 100  0

user1@vm1:~$ vmstat -S M -a
#                     (17)    (18)
procs -----------memory---------- ---swap-- -----io---- -system-
- ----cpu----
 r  b   swpd   free  inact active   si   so    bi    bo   in   c
s us sy id wa
 0  0      0     11    434     19    0    0    24     2   11
6  0  0 100  0

user1@vm1:~$ vmstat -d
#19    (20)    (21)    (22)    (23)  (24)    (25)    (26)    (27
)   (28)    (29)
disk- ------------reads------------ ------------writes----------
- -----IO------
       total merged sectors      ms  total merged sectors      m
s    cur    sec
sda    11706    353  402980   17612   9303  40546  336358   4698
0      0     19
sr0        0      0       0       0      0      0       0
0      0      0
loop0      0      0       0       0      0      0       0
0      0      0

user1@vm1:~$ vmstat -m | head
#(30)                        (31)  (32)   (33)   (34)
Cache                        Num  Total   Size  Pages
ext3_inode_cache           13700  13700    808     10
ext3_xattr                     0      0     88     46
journal_handle               170    170     24    170
journal_head                  37     72    112     36
revoke_table                 256    256     16    256
revoke_record                128    128     32    128
kmalloc_dma-512                8      8    512      8
ip6_dst_cache                 16     24    320     12
UDPLITEv6                      0      0   1024      8
```

字段和描述：

模式	情况	字段	描述
虚拟内存	进程	（1）	r：等待运行的进程数。

		（2）	b：不间断睡眠中的进程数。
	内存	（3）	swpd：使用的虚拟内存量。
		（4）	free：空闲内存量。
		（5）	buff：用作缓冲区的内存量。
		（6）	cache：用作缓存的内存量。
		（17）	inact：非活动内存量。
		（18）	active：活动内存量。
	交换	（7）	si：从磁盘换入的内存量（/秒）。
		（8）	so：换出到磁盘的内存量（/秒）。
	I/O	（9）	bi：从设备接收的块（块/秒）。
		（10）	bo：发送到设备的块（块/秒）。
	系统	（11）	in：每秒中断的次数，包括时钟。
		（12）	cs：每秒上下文切换的数量。
	CPU	（13）	us：运行非内核代码的时间。（用户时间，包括优先的时间）
		（14）	sy：运行内核代码的时间。（系统时间）
		（15）	id：闲置时间。在 Linux 2.5.41 之前，这包括 IO 等待时间。
		（16）	wa：IO 等待时间。在 Linux 2.5.41 之前，包含在闲置时间中。
磁盘，`-d`	设备	（19）	设备名称
	读	（20）	total：成功完成的总读取数
		（21）	merge：分组的读取数（生成一个 I/O）
		（22）	sectors：成功读取的分区
		（23）	ms：用于读取的毫秒
	写	（24）	total：成功完成的总写入数
		（25）	merge：分组的写入数（生成一个 I/O）
		（26）	sectors：成功写入的分区
		（27）	ms：用于写入的毫秒
	I/O	（28）	cur：正在进行中的 I/O
		（29）	s：用于 I/O 的秒数

Slab，`-m`	Slab	（30）	缓存：缓存名称
		（31）	num：当前活动对象的数量
		（32）	total：可用对象的总数
		（33）	size：每个对象的大小
		（34）	page：具有至少一个活动对象的页数

`top` 的输出：

```
#     (1)      (2)                (3)      (4)
top - 03:22:44 up 4 days, 22:54,  1 user,  load average: 0.00, 0
.00, 0.00
#       (5)         (6)         (7)            (8)          (9)
Tasks:  63 total,   1 running,  62 sleeping,   0 stopped,   0 zo
mbie
#       (10)     (11)     (12)    (13)      (14)     (15)     (
16)     (17)
Cpu(s):  0.0%us,  1.1%sy,  0.0%ni, 98.9%id,  0.0%wa,  0.0%hi,  0
.0%si,  0.0%st
#                (18)            (19)            (20)
 (21)
Mem:    508820k total,   273792k used,   235028k free,    27844k
 buffers
#                (22)            (23)            (24)
 (25)
Swap:   473080k total,        0k used,   473080k free,   201252k
 cached

#(26) (27)     (28)(29) (30) (31) (32,33) (34)(35)      (36) (37
)
  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COM
MAND
    1 root      20   0  8356  804  676 S  0.0  0.2   0:05.99 ini
t
    2 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kth
readd
    3 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 mig
ration/0
    4 root      20   0     0    0    0 S  0.0  0.0   0:00.06 kso
ftirqd/0
    5 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 wat
chdog/0
    6 root      20   0     0    0    0 S  0.0  0.0   0:03.25 eve
nts/0
    7 root      20   0     0    0    0 S  0.0  0.0   0:00.00 cpu
set
<...>
```

字段和输出：

部分	字段	描述
正常运行时间	（1）	当前时间。
	（2）	正常运行时间（启动后的时间）。
	（3）	目前有多少用户登录。
	（4）	过去 1，5 和 15 分钟内的 CPU 负载。这不是规范化的，所以负载均值为 1 意味着单个 CPU 的满负载，但是在 4 个 CPU 的系统上，这意味着它有 75% 空闲时间。
任务	（5）	运行进程总数。
	（6）	当前正在执行的进程数。
	（7）	当前正在睡眠的进程数。
	（8）	被停止的进程数（例如使用 CTRL + Z ）。
	（9）	已经停止（“僵尸”）的进程数量，已终止，但未由其父进程回收。
CPU（S）	（10）	CPU 运行不优先的用户进程的时间。
	（11）	CPU 运行内核及其进程的时间。
	（12）	CPU 运行优先的用户进程的时间。
	（13）	CPU 花费的空闲时间。
	（14）	CPU 等待 I/O 完成的时间。
	（15）	CPU 维护硬件中断的时间。
	（16）	CPU 维护软件中断的时间。
	（17）	由管理程序从这个虚拟机“偷走”的 CPU 总量，用于其他任务（例如启动另一个虚拟机）。
内存/交换	（18）	物理内存总量。
	（19）	使用的物理内存总量。
	（20）	完全空闲的物理内存。
	（21）	专用于缓存磁盘块的 RAM 和文件系统元数据总量。
	（22,23,24）	总，使用和空闲交换内存。
	（25）	专用于从文件读取的页面的 RAM 总量。
进程	（26）	任务的唯一进程 ID，它定期地包装，尽管从未重新启动。

	（27）	任务所有者的有效用户名。
	（28）	任务的优先级。
	（29）	任务的优先值。负的优先值表示更高的优先级，而正的优先值表示较低的优先级。在这个字段中的零只是代表在确定任务的调度时不会调整优先级。
	（30）	任务使用的虚拟内存总量。它包括所有代码，数据和共享库，以及已经被替换的页面。以及已被映射但未被使用的页面。
	（31）	任务已使用的未交换的物理内存。
	（32）	任务使用的共享内存量。它只是反映可能与其他进程共享的内存。
	（33）	任务的状态可以是以下之一：`D` =不间断睡眠，`R` =运行，`S` =睡眠，`T` =跟踪或停止，`Z` =僵尸。
	（34）	自上次屏幕更新以来，所经过的 CPU 时间的任务份额，以 CPU 时间总数的百分比表示。
	（35）	任务当前使用的可用物理内存的份额。
	（36）	CPU 时间，单位是百分之一秒，与 `TIME` 相同，但通过百分之一秒反映更大的粒度。
	（37）	命令 - 命令行或程序名称

你可能会看到很多字段。许多字段都存在于多个工具中，这些工具有些冗余的功能。通常情况下，你只需要这个字段的一小部分，但你需要知道，系统性能的许多信息（实际上还有更多）可用于你，因为有时候会出现一个模糊的问题，并且为了能够解决它，需要知道如何读取这些数据。

现在，你将学习如何使用系统性能工具。

这样做

```
 1: uptime
 2: free
 3: vmstat
 4: ( sleep 5 && dd if=/dev/urandom of=/dev/null bs=1M count=30
&& sleep 5 && killall vmstat )& vmstat 1
 5: uptime
 6: ( sleep 5 && dd if=/dev/zero of=test.img bs=32 count=$((32*1
024*200)) && sleep 5 && killall vmstat )& vmstat -nd 1 | egrep -
v 'loop|sr0'
 7: echo 3 | sudo tee /proc/sys/vm/drop_caches
 8: free -mt ; find / >/dev/null 2>&1 ; free -mt
 9: echo 3 | sudo tee /proc/sys/vm/drop_caches
10: cat test.img /dev/null ; free -mt
```

你会看到什么

```
user1@vm1:~$ uptime
 05:36:45 up 6 days,  1:08,  1 user,  load average: 0.00, 0.00,
0.00
user1@vm1:~$ free
             total       used       free     shared    buffers
   cached
Mem:        508820     239992     268828          0        820
   213720
-/+ buffers/cache:      25452     483368
Swap:       473080          0     473080
user1@vm1:~$ vmstat
procs -----------memory---------- ---swap-- -----io---- -system-
- ----cpu----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   c
s us sy id wa
 0  0      0 268828    820 213720    0    0    21    10   14   1
1  0  0 100  0
user1@vm1:~$ ( sleep 5 && dd if=/dev/urandom of=/dev/null bs=1M
count=30 && sleep 5 && killall vmstat )& vmstat 1
[1] 6078
procs -----------memory---------- ---swap-- -----io---- -system-
- ----cpu----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   c
s us sy id wa
 1  1      0 268556    828 213736    0    0    21    10   14   1
1  0  0 100  0
 0  0      0 268556    828 213772    0    0    16     0   19   1
0  0  0 100  0
 0  0      0 268556    828 213772    0    0     0     0   13
8  0  0 100  0
 0  0      0 268556    828 213772    0    0     0     0   15   1
1  0  0 100  0
 0  0      0 268556    828 213772    0    0     0     0   14   1
```

```
0  0  0 100  0
 0  0      0 268556    828 213772    0    0     0     0   18   1
3  0  0 100  0
 1  0      0 267316    836 213844    0    0    74     0  267   2
6  0 99  1  0
 1  0      0 267316    836 213844    0    0     0     0  303
7  0 100  0  0
 1  0      0 267316    836 213844    0    0     0     0  271   1
1  0 100  0  0
 1  0      0 267316    836 213844    0    0     0     0  257   1
2  0 100  0  0
30+0 records in
30+0 records out
31457280 bytes (31 MB) copied, 4.95038 s, 6.4 MB/s
 0  0      0 267928    860 213860    0    0    27     0  265   2
9  1 97  2  0
 0  0      0 267936    860 213848    0    0     0     0   15
9  0  0 100  0
 0  0      0 267936    860 213848    0    0     0     0   14
7  0  0 100  0
 0  0      0 267936    860 213848    0    0     0     0   14
7  0  0 100  0
 0  0      0 267936    860 213848    0    0     0     0   13   1
1  0  0 100  0
Terminated
user1@vm1:~$ uptime
 05:22:15 up 6 days, 54 min,  1 user,  load average: 0.07, 0.02,
 0.00
[1]+  Done                    ( sleep 5 && dd if=/dev/urandom of
=/dev/null bs=1M count=30 && sleep 5 && killall vmstat )
user1@vm1:~$ uptime
 05:22:22 up 6 days, 54 min,  1 user,  load average: 0.06, 0.02,
 0.00
user1@vm1:~$ ( sleep 5 && dd if=/dev/zero of=test.img bs=32 coun
t=$((32*1024*200)) && sleep 5 && killall vmstat )& vmstat -nd 1
| egrep -v 'loop|sr0'
[1] 6086
disk- ------------reads------------ ------------writes----------
- -----IO------
       total merged sectors      ms  total merged sectors      m
s    cur    sec
sda   146985 2230744 21821320  105848  32190 1343154 10927338 13
30144      0    105
sda   146995 2230744 21821648  105848  32190 1343154 10927338 13
30144      0    105
sda   146995 2230744 21821648  105848  32190 1343154 10927338 13
30144      0    105
sda   146995 2230744 21821648  105848  32190 1343154 10927338 13
30144      0    105
sda   146995 2230744 21821648  105848  32190 1343154 10927338 13
30144      0    105
sda   146995 2230744 21821648  105848  32190 1343154 10927338 13
30144      0    105
```

```
sda   146999 2230744 21821680  105856  32190 1343154 10927338 13
30144      0    105
sda   146999 2230744 21821680  105856  32190 1343154 10927338 13
30144      0    105
sda   147000 2230744 21821688  105856  32208 1344160 10935530 13
30288      0    105
sda   147000 2230744 21821688  105856  32274 1349214 10976490 13
30748      0    105
sda   147000 2230744 21821688  105856  32325 1353259 11009258 13
31236      0    105
sda   147000 2230744 21821688  105856  32450 1364657 11101442 13
37176      0    105
sda   147000 2230744 21821688  105856  32450 1364657 11101442 13
37176      0    105
sda   147001 2230744 21821696  105856  32471 1366301 11114762 13
37348      0    105
sda   147001 2230744 21821696  105856  32525 1370529 11149018 13
37732      0    105
sda   147001 2230744 21821696  105856  32573 1374577 11181786 13
38064      0    105
sda   147001 2230744 21821696  105856  32698 1386562 11278666 13
46244      0    105
6553600+0 records in
6553600+0 records out
209715200 bytes (210 MB) copied, 11.7088 s, 17.9 MB/s
sda   147001 2230744 21821696  105856  32698 1386562 11278666 13
46244      0    105
sda   147001 2230744 21821696  105856  32698 1386562 11278666 13
46244      0    105
sda   147001 2230744 21821696  105856  32698 1386562 11278666 13
46244      0    105
sda   147001 2230744 21821696  105856  32698 1386562 11278666 13
46244      0    105
sda   147001 2230744 21821696  105856  32762 1393910 11337962 13
49192      0    105
user1@vm1:~$ echo 3 | sudo tee /proc/sys/vm/drop_caches
3
[1]+  Done                    ( sleep 5 && dd if=/dev/zero of=te
st.img bs=32 count=$((32*1024*200)) && sleep 5 && killall vmstat
 )
user1@vm1:~$ free -mt ; find / >/dev/null 2>&1 ; free -mt
             total       used       free     shared    buffers
   cached
Mem:           496         30        466          0          0
        5
-/+ buffers/cache:         24        472
Swap:          461          0        461
Total:         958         30        928
             total       used       free     shared    buffers
   cached
Mem:           496         64        432          0         22
        6
-/+ buffers/cache:         35        461
```

```
Swap:          461          0        461
Total:         958         64        894
user1@vm1:~$ echo 3 | sudo tee /proc/sys/vm/drop_caches
3
user1@vm1:~$ cat test.img /dev/null ; free -mt
             total       used       free     shared    buffers
   cached
Mem:           496        230        265          0          0
      205
-/+ buffers/cache:         24        471
Swap:          461          0        461
Total:         958        230        727
user1@vm1:~$
```

解释

1. 打印当前的正常运行时间。
2. 打印出可用内存信息。
3. 这个很有趣，最好认为是一种实验。首先，我们在后台启动 `( sleep 5 && dd if=/dev/urandom of=/dev/null bs=1M count=30 &&` ，之后我们 以连续模式启动 `vmstat` ，所以它将打印出其信息直到中断。我们可以看到，在这个命令启动 5 秒钟后，CPU 负载显着增加了一段时间，然后减少，另外 5 秒钟后 `vmstat` 被杀死。
4. 打印当前的正常运行时间。注意负载平均值的变化。
5. 这是另一个实验，几乎和以前一样，但这次用磁盘写入。
6. 删除所有缓存和缓冲区。
7. 另一个实验。我们想看看读取系统中的所有文件和目录名称，会如何影响内存中的文件系统缓存，并且我们可以看到它被缓存在缓冲区中，这是有理论根据的。
8. 再次删除所有缓存和缓冲区。
9. 这次我们想看看，文件读取如何影响内存中的文件系统缓存。我们可以看到读取的文件被缓存在缓存部分，来增加后续访问的时间。

附加题

- 为什么在我们的第一个实验中，不是 `user` ，而是 `system` CPU 使用率上升到 100？
- 这是什么意思？ `dd if=/dev/zero of=test.img bs=32 count=$( (32*1024*200) )` 。
- 启动 `top` ，并按下 `h` 。现在尝试按照 CPU，内存和 PID 对其输出进行排序。

练习 29：内核：内核消息，`dmesg`

> 原文：Exercise 29. Kernel: kernel messages, dmesg
>
> 译者：飞龙
>
> 协议：CC BY-NC-SA 4.0
>
> 自豪地采用谷歌翻译

那么，如果你到达了这里，现在是谈谈内核的时候了。我们将使用维基百科的操作系统内核定义，开始这个讨论：

> 在计算机中，内核（来自德语 Kern）是大多数计算机操作系统的主要组成部分；它是应用程序和硬件级别上进行的实际数据处理之间的桥梁。内核的职责包括管理系统的资源（硬件和软件组件之间的通信）。通常，作为操作系统的基本组件，内核可以为资源（特别是处理器和 I/O 设备）提供最底层的抽象，应用软件必须控制它来执行其功能。它通常通过进程间通信机制和系统调用，使这些设施可用于应用程序进程。

这是维基百科告诉我们的，Linux 内核的具体内容：

> Linux 内核是 Linux 系列类 Unix 操作系统使用的操作系统内核。它是自由和开源软件最突出的例子之一。它支持真正的抢占式多任务（在用户模式和内核模式下），虚拟内存，共享库，按需加载，共享的写时复制（COW）可执行文件，内存管理，互联网协议组和线程。

现在是访问相应的维基百科文章的好时机，并花费一些时间疯狂点击所有可怕的术语，它们描述 Linux 内核的技术特性。这样做之后，让我们谈谈更多的单调的主题，这是内核告诉我们的一种方式。例如，如果 USB 记忆棒连接到计算机，或者网络链接断开或挂载了文件系统，则会发生这种情况。为了能够告诉你所有这些东西，内核使用一种称为显示消息 或驱动消息的机制，其名称缩写为 `dmesg` 。

该机制由固定大小的缓冲区表示，内核向它写入消息。在 Debian Linux 上，系统日志守护进程启动后，从缓冲区发布的信息也会被复制到 `/var/log/dmesg` 。这样做是为了保留这些消息，否则将被新的消息覆盖。

`dmesg` 也是工具的名称，它允许你查看当前在内核缓冲区中的那些消息，并更改此缓冲区大小。

让我总结一下 `dmesg` 相关的文件和程序：

- `dmesg` - 打印或控制内核环缓冲区
- `/var/log/dmseg` - Debian 发行版中的日志文件，仅包含系统引导期间的 `dmesg` 消息副本，而不包含时间戳。
- `/var/log/kern.log` - Debian 发行版中的日志文件，包含所有 `dmesg` 消息的副本，包括时间戳请注意， `rsyslog` 日志守护进程启动后，这个时间戳开始变化，这意味着 `rsyslog` 启动前，所有引导时的消息将具有相同的时间

戳。此文件本身包含 `/var/log/dmseg` 。

- `/var/log/messages` - Debian 发行版中的日志文件，记录所有非调试和非关键消息。它本身包含 `/var/log/dmesg` 。
- `/var/log/syslog` - Debian 发行版中的日志文件，记录了所有信息，但权限相关的信息除外。它包含 `/var/log/messages` 和 `/var/log/kern.log` 中的所有消息。

这样做

```
1: date
2: sudo umount /tmp ; sudo mount /tmp
3: sudo tail -f /var/log/dmesg /var/log/messages /var/log/syslog
 /var/log/kern.log
```

你会看到什么

```
user1@vm1:~$ date
Tue Jul 24 06:55:33 EDT 2012
user1@vm1:~$ sudo umount /tmp ; sudo mount /tmp
user1@vm1:~$ dmesg | tail
[    7.166240] tun: Universal TUN/TAP device driver, 1.6
[    7.166242] tun: (C) 1999-2004 Max Krasnyansky <maxk@qualcomm
.com>
[    7.432019] ADDRCONF(NETDEV_UP): eth0: link is not ready
[    7.435270] e1000: eth0 NIC Link is Up 1000 Mbps Full Duplex,
 Flow Control: RX
[    7.435927] ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
[   17.472049] tap0: no IPv6 routers present
[   17.592044] eth0: no IPv6 routers present
[  217.497357] kjournald starting.  Commit interval 5 seconds
[  217.497561] EXT3 FS on sda8, internal journal
[  217.497564] EXT3-fs: mounted filesystem with ordered data mod
e.
user1@vm1:~$ sudo tail /var/log/dmesg /var/log/messages /var/log
/syslog /var/log/kern.log
==> /var/log/dmesg <==
[    6.762569] EXT3 FS on sda5, internal journal
[    6.762572] EXT3-fs: mounted filesystem with ordered data mod
e.
[    6.767237] kjournald starting.  Commit interval 5 seconds
[    6.767407] EXT3 FS on sda6, internal journal
[    6.767410] EXT3-fs: mounted filesystem with ordered data mod
e.
[    7.166240] tun: Universal TUN/TAP device driver, 1.6
[    7.166242] tun: (C) 1999-2004 Max Krasnyansky <maxk@qualcomm
.com>
[    7.432019] ADDRCONF(NETDEV_UP): eth0: link is not ready
```

```
[    7.435270] e1000: eth0 NIC Link is Up 1000 Mbps Full Duplex,
 Flow Control: RX
[    7.435927] ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready

==> /var/log/messages <==
Jul 24 06:52:07 vm1 kernel: [    6.767407] EXT3 FS on sda6, inte
rnal journal
Jul 24 06:52:07 vm1 kernel: [    6.767410] EXT3-fs: mounted file
system with ordered data mode.
Jul 24 06:52:07 vm1 kernel: [    7.166240] tun: Universal TUN/TA
P device driver, 1.6
Jul 24 06:52:07 vm1 kernel: [    7.166242] tun: (C) 1999-2004 Ma
x Krasnyansky <maxk@qualcomm.com>
Jul 24 06:52:07 vm1 kernel: [    7.432019] ADDRCONF(NETDEV_UP):
eth0: link is not ready
Jul 24 06:52:07 vm1 kernel: [    7.435270] e1000: eth0 NIC Link
is Up 1000 Mbps Full Duplex, Flow Control: RX
Jul 24 06:52:07 vm1 kernel: [    7.435927] ADDRCONF(NETDEV_CHANG
E): eth0: link becomes ready
Jul 24 06:55:36 vm1 kernel: [  217.497357] kjournald starting.
Commit interval 5 seconds
Jul 24 06:55:36 vm1 kernel: [  217.497561] EXT3 FS on sda8, inte
rnal journal
Jul 24 06:55:36 vm1 kernel: [  217.497564] EXT3-fs: mounted file
system with ordered data mode.

==> /var/log/syslog <==
Jul 24 06:52:08 vm1 acpid: 1 rule loaded
Jul 24 06:52:08 vm1 acpid: waiting for events: event logging is
off
Jul 24 06:52:08 vm1 /usr/sbin/cron[882]: (CRON) INFO (pidfile fd
 = 3)
Jul 24 06:52:08 vm1 /usr/sbin/cron[883]: (CRON) STARTUP (fork ok
)
Jul 24 06:52:08 vm1 /usr/sbin/cron[883]: (CRON) INFO (Running @r
eboot jobs)
Jul 24 06:52:16 vm1 kernel: [   17.472049] tap0: no IPv6 routers
 present
Jul 24 06:52:16 vm1 kernel: [   17.592044] eth0: no IPv6 routers
 present
Jul 24 06:55:36 vm1 kernel: [  217.497357] kjournald starting.
Commit interval 5 seconds
Jul 24 06:55:36 vm1 kernel: [  217.497561] EXT3 FS on sda8, inte
rnal journal
Jul 24 06:55:36 vm1 kernel: [  217.497564] EXT3-fs: mounted file
system with ordered data mode.

==> /var/log/kern.log <==
Jul 24 06:52:07 vm1 kernel: [    7.166240] tun: Universal TUN/TA
P device driver, 1.6
Jul 24 06:52:07 vm1 kernel: [    7.166242] tun: (C) 1999-2004 Ma
x Krasnyansky <maxk@qualcomm.com>
Jul 24 06:52:07 vm1 kernel: [    7.432019] ADDRCONF(NETDEV_UP):
```

```
eth0: link is not ready
Jul 24 06:52:07 vm1 kernel: [    7.435270] e1000: eth0 NIC Link
is Up 1000 Mbps Full Duplex, Flow Control: RX
Jul 24 06:52:07 vm1 kernel: [    7.435927] ADDRCONF(NETDEV_CHANG
E): eth0: link becomes ready
Jul 24 06:52:16 vm1 kernel: [   17.472049] tap0: no IPv6 routers
 present
Jul 24 06:52:16 vm1 kernel: [   17.592044] eth0: no IPv6 routers
 present
Jul 24 06:55:36 vm1 kernel: [  217.497357] kjournald starting.
Commit interval 5 seconds
Jul 24 06:55:36 vm1 kernel: [  217.497561] EXT3 FS on sda8, inte
rnal journal
Jul 24 06:55:36 vm1 kernel: [  217.497564] EXT3-fs: mounted file
system with ordered data mode.
```

解释

1. 打印出当前日期和时间。
2. 从内核消息缓冲区打印最后 10 条消息。
3. 从 `/var/log/dmesg` , `/var/log/messages` , `/var/log/syslog` 和 `/var/log/kern.log` 打印最后 10 条消息。

附加题

这就完了，没有附加题，哇哦！

练习 30：打磨、洗练、重复：总复习

> 原文：Exercise 30. Lather, Rinse, Repeat: The Grand Rote Learning
>
> 译者：飞龙
>
> 协议：CC BY-NC-SA 4.0
>
> 自豪地采用谷歌翻译

本指南中的信息量相当大。没有足够长的练习和一些深入研究，你不能记住它。所以剩下的唯一的工作就是填写这张表，每天都把这张表打印在你的记忆中，直到你知道了它。

你可能想问，为什么需要记住所有这些东西，如果你可以随时查看的话。那么简短的答案是因为你不能。这意味着为了高效地查找事物，你需要知道要寻找什么，并且为了知道要寻找什么，你需要一个坚实的基础。一旦你有了这个基础，一旦你明白什么是重要的，什么不是，以及系统的组织方式，你将能够高效寻找东西。

你可能会想知道，为什么在我的指南中有很多详细的表格，其中包含许多字段的列表，其中包含几乎不需要的信息。你必须明白的是，你应该以这种方式训练自己，来查看任何控制台程序。你应该熟悉这个信息，而不像是一本科幻小说那样，其中你可能不会注意细节，但仍然很了解它。你应该将所有这些数据看做数学公式，其中每个符号都有其意义，甚至更多，如果你不明白特定的符号意味着什么，你将无法走地更远。

有时完全可以留下一些未解释的东西，但让自己变得更深入，即使经常是这样。通过研究这个特定的工具，了解它告诉你什么以及为什么，给自己一个礼物。如果你这样做，如果你会深入内部，你对操作系统的理解（在我们这种情况下是 Linux）将会极大增加。

文档

`man`，`info`

命令或概念	含义
`man`	
`info`	
`man 1`	
`man 2`	
`man 3`	
`man 4`	
`man 5`	
`man 6`	
`man 7`	
`man 8`	
`man 9`	
`man -k`	
`man -wK`	
粗体	
斜体	
`[]`	
`-a|-b`	
`argument ...`	
`[expression] ...`	

Google 和实用资源

搜索术语/资源	含义
`(a|b) c`	
`site:foo.bar`	
`"a long query"`	
http://en.wikipedia.org	
http://stackexchange.com/	
http://www.cyberciti.biz/	
http://tldp.org/	
`programname.site`	

包管理：Debian 包管理工具 `aptitude`

命令或概念	含义
`aptitude`	
`aptitude search`	
`aptitude install`	
`dpkg -l`	
`dpkg -L`	
预期操作	
包状态	
http://www.debian.org/distrib/packages	

系统启动：运行级别，`/etc/init.d`，`rcconf`，`update-rc.d`

命令或概念	含义
`rcconf`	
`update-rc.d`	
`sysv-rc-conf`	
运行级别	
运行级别 1	
运行级别 2	
运行级别 6	

进程：处理进程，`ps`，`kill`

命令或概念	含义
`ps`	
`kill`	
`ps ax`	
`ps axue`	
`ps axue --forest`	
信号	
`HUP`	
`TERM`	
`KILL`	
为什么 `KILL -9` 是不好的？	

任务调度：`cron`，`at`

命令或概念	含义
`crontab -l`	
`crontab -e`	
`crontab -r`	
`crontab /foo`	
`crontab > foo`	
`* * * * *`	
`at`	
`atq`	
`atq`	
`atrm`	
`batch`	

日志，`/var/log`，`rsyslog`，`logger`

命令或概念	含义
`logger`	
`grep -irl`	
`find . -mmin -5`	
`tail -f`	
`logrotate`	
日志守护程序	
日志级别	
日志轮替	

文件系统

命令或概念	含义
文件系统	
文件	
目录	
索引节点	
块	
挂载	
UUID	
日志	
MBR	
分区	
分区表	

挂载，`mount`，`/etc/fstab`

命令或概念	含义
`parted`	
`cfdisk`	
`fdisk`	
`mount`	
`umount`	
`mount -a`	
`/etc/fstab`	
`fsck`	
`blkid`	

创建和修改文件系统，`mkfs`，`tune2fs`

命令或概念	含义
`tune2fs`	
`mkfs`	
块大小	
保留块数量	
最大挂载数量	
检查间隔	

更改根目录，`chroot`

命令或概念	含义
`chroot`	
`ldd`	
根目录	
更改根目录	
动态库依赖	

移动数据：`tar`，`dd`

命令或概念	含义
`tar`	
`dd`	
`losetup`	

安全权限：`chown`，`chmod`

命令或概念	含义
`chmod`	
`chown`	
`umask`	
权限	
权限模式	
权限类	
Umask 机制	

网络

网络概念	含义
OSI 模型	
DOD 模型	
通信协议	
以太网	
MAC 地址	
以太网广播地址	
TCP/IP	
IP	
IP 封包	
IP 地址	
IP 子网	
端口	
网络套接字	
本地套接字地址	
远程套接字地址	
套接字对	
路由	
默认网关	
IP 广播地址	
ICMP	
TCP	
TCP 封包	
UDP	
UDP 封包	
主机名称	

网络配置，`ifconfig`，`netstat`，`iproute2`，`ss`

命令或概念	含义
`/etc/network/interfaces`	
`auto`	
`allow-hotplug`	
`/etc/hosts`	
`/etc/hostname`	
`localhost`	
回送接口	
伪接口	

封包过滤配置，`iptables`

命令或概念	含义
`iptables-save`	
`iptables`	
`modprobe`	
`nc`	
`tcpdump`	
`LINKTYPE_LINUX_SLL`	
以太网帧头部	
IPv4 头部	
TCP 段	
`netfilter`	
iptables 表	
iptables 链	
iptables 目标	

安全 Shell, `ssh` ，`sshd` ，`scp`

命令或概念	含义
`ssh`	
`sshd`	
`scp`	
`ssh-keygen`	
主机密钥	
证密钥	
数据加密密码	
数据完整性算法	
SSH 会话密钥	

性能：获取性能状态，`uptime` ，`free` ，`top`

命令或概念	含义
`uptime`	
`free`	
`vmstat`	
`top`	
CPU 占用（`us`，`sy`，`id`，`wa`）	
内存（`swpd`，`free`，`buff`，`cache`，`inact`，`active`）	
Slab 分配	
磁盘（`IOPS`，`read`，`write`）	
进程（`PR`，`NI`，`VIRT`，`RES`，`SHR`，`Status`）	

内核：内核消息，`dmesg`

命令或概念	含义
`dmseg`	
`/var/log/dmesg`	
`/var/log/messages`	
`/var/log/syslog`	
`/var/log/kern.log`	
内核消息缓冲区	

下一步做什么

> 原文：What to do next
>
> 译者：飞龙
>
> 协议：CC BY-NC-SA 4.0
>
> 自豪地采用谷歌翻译

恭喜你到达了这里，但你的旅程才刚刚开始。请参阅下面的资源，来了解之后要做什么。

- 每天阅读一个手册页。使其成为习惯。每天阅读一个随机的手册页。我的笔记本上现在有大约 6000 个手册页，所以可以看很多年。
- 从零开始构建你自己的 Linux 发行版：http://www.linuxfromscratch.org/lfs/。你可能希望将我的 Debian 装置用于此任务和其他任务。
- 自己学一些正则表达式：http://regex.learncodethehardway.org/
- 自己学一些 bash 脚本：http://mywiki.wooledge.org/BashGuide
- 看书。例如，这本不错：《Unix 和 Linux管理手册》。另请阅读[《Unix 厌恶者手册》](http://en.wikipedia.org/wiki/The_UNIX-HATERS_Handbook，并写出你认为仍然有效的那些观点。现在意识到，所有的操作系统都是糟糕的，只是有些比其它更糟糕。
- 去找一个提供 VPS（虚拟专用服务器或虚拟机）的托管服务器。安装像 Apache 这样的东西和你自己的 wiki。在线记录你的发现。
- 请访问 http://technet.microsoft.com/en-us/virtuallabs/bb467605.aspx，看看 Microsoft 技术可以做什么。
- 在你的 VPS 上按照 https://help.ubuntu.com/community/Servers 设置一切。只需设置，检查它是否正常工作并将其删除。或不要删除。无论如何，它将为你提供服务器管理所需的经验。

我会在某一天把它做得更好，但是由于这个资源列表，你应该已经很忙了。祝你好运。